UNIVERSIDAD NACIONAL

Facultad de Ciencias Exactas y Naturales

ESCUELA DE INFORMÁTICA

“PROPUESTA DE IMPLEMENTACIÓN DE UN FRAMEWORK DE AUTOMATIZACIÓN DE PRUEBAS PARA ICOST EN COMPONENTES INTEL DE COSTA RICA”

Para optar al grado de Licenciado en Informática

con énfasis en Sistemas Web

Ing. Mariela Itzel Sequeira Ugalde

Ing. Juan Diego González Arias

Heredia, Costa Rica

Agradecimientos y dedicatorias

Este proyecto es dedicado a mi familia, en especial a mi esposo Daniel Rodríguez, a mi mamá Guiselle Ugalde, a mi papá Ronald Sequeira, a mi hermana Sofia Sequeira y a mi abuela Ruth Varela, que siempre han estado presentes para motivarme a continuar con mis estudios.

Quiero expresar mi agradecimiento a nuestro tutor Cristopher Montero por acompañarnos a lo largo del proyecto, por su paciencia, por corregirnos y por motivarnos en todo momento.

Le agradezco también a nuestro patrocinador, Intel, en especial a Soledad Riggioni quien siempre nos apoyó y nos brindó todos los recursos necesarios para completar este proyecto.

Mariela Sequeira Ugalde

Este proyecto se lo dedico a mi mamá Ruth Arias Araya que me ha apoyado en toda mi vida y siempre se ha esforzado para sacarme adelante en mis estudios.

Agradecimientos a Cristopher Montero por aceptar ser nuestro tutor y brindarnos guía durante este trayecto, también agradecer al personal de ICOST por brindarnos ayuda y estar disponibles cuando se les necesitó y a Intel por permitirnos realizar nuestro proyecto dentro de la empresa.

Juan Diego González Arias

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO I
INTRODUCCIÓN

CAPÍTULO I: INTRODUCCIÓN

1. Antecedentes

Intel Corporation (Integrated Electronic) es una corporación multinacional estadounidense que cuenta con sedes alrededor del mundo entre estas incluyen a Costa Rica, Intel fue pionero en el mundo de los microprocesadores, aportando los primeros microprocesadores en el mercado, actualmente es uno de los mayores fabricantes de microprocesadores proveyendo a empresas como Apple, Lenovo, Dell, Hewlett Packard, etc.

Intel fue fundada por Gordon E. Moore en 1968, creció en el negocio de los electrónicos con circuitos lógicos y con la meta del mercado de las memorias semiconductoras, pronto creó el primer microprocesador que daría campo a una nueva era y a lo que es hoy en día la tecnología y encontrando a su mayor competidor en la actualidad el cual es AMD.

Además de estar en el mercado de los microprocesadores Intel ha ingresado en otros mercados como la Nube y los centros de datos, el almacenamiento de la información, otros serían la seguridad informática, el internet en las cosas, dispositivos inteligentes (smart devices) e inteligencia artificial.

Intel Costa Rica abrió por primera vez en el año 1997, actualmente es de los mayores centros de investigación y desarrollo (I+D) y de los mejores centros de servicio en el país que empezaron a operar en el 2014, actualmente se cuenta con más de 2000 empleados que se dedican a funciones de testeo, validación de circuitos integrados, diseño, prototipado y soluciones de software, por supuesto también cuenta con su parte de recursos humanos, finanzas, ventas y mercado y TI.

- Misión: “Utilizar el poder de la ley de Moore para llevar dispositivos inteligentes y conectados a cada persona en la Tierra”.

- Visión: “Si es inteligente y conectado, es mejor con Intel”.

Icost es el departamento en donde se va a localizar este proyecto, además es el departamento encargado de la gestión de los inventarios de Intel, para esto el mismo departamento maneja una herramienta que de igual manera se llama ICOST, a partir de este momento llamaremos de esta manera al software utilizado en este proyecto, es una herramienta integrada para costos de venta, además maneja el pronósticos de costos y funciones de inventario por medio de un portal web, es una aplicación que maneja una gran cantidad de datos puesto que se integra con más de 60 aplicaciones internas de Intel, sitios y fábricas.

Como se sabe el manejo de datos sobre inventarios y costos es uno de los temas más importantes en una empresa, ICOST es el departamento que se encarga del desarrollo de herramienta para realizar el manejo de Intel sobre costos e inventarios, por lo tanto, este proyecto a realizar viene a brindar una solución y un apoyo a este departamento.

Hasta el mes de Abril del 2018 el departamento de ICOST utilizaba un software llamado Legacy ICOST, el cual llevaba el manejo de inventario tiempo atrás, pero al ser un software desarrollado hace varios años, la tecnología con la que contaba estaba ya desfasada, lo cual llevó al departamento ICOST a optar por el desarrollo de una nueva herramienta en la cual se pudiera utilizar y combinar con nuevas tecnologías que permitieran una mejor ejecución del proceso de manejo de inventarios y reportes.

Anteriormente se inició la automatización de Legacy ICOST, se crearon pocas automatizaciones, pero ninguna de estas se creó bajo ningún formato, estándar o marco de trabajo, por lo cual tampoco era un trabajo rápido la creación de procesos automatizados y el mantenimiento no era un proceso fácil.

La herramienta de automatización utilizada para Legacy ICOST es llamada Worksoft Certify, esta se especializa en la automatización de procesos en SAP, páginas web y móviles, lo que permitió su uso en Legacy ICOST para los procesos de automatización, ya que era una herramienta web, además de que Intel ya contaba con licencias para el uso de Certify.

Luego de la implementación del nuevo sitio ICOST Evolve, se dejó de utilizar Legacy ICOST lo que provocó que la automatización existente hasta el momento quedara obsoleta, pues no era adaptable a la nueva aplicación, lo que provocó un retraso a la hora de realizar las pruebas ya que éstas deben de ser ejecutadas manualmente por analistas.

2. Planteamiento del problema

Actualmente ICOST cuenta con aproximadamente 80 pantallas y 60 reportes por lo que podría considerarse como una herramienta robusta, en la cual fluyen datos de diversas aplicaciones de la empresa y se encuentra en constante actualización, se le realizan mejoras y corrigen errores, y cada uno de estos debe ser probado para que el cambio pueda ser reflejado en el ambiente de producción, y cada una de estas pruebas tiene una duración de aproximadamente 2 horas, lo cual representa una alta demanda de recursos para realizar estos escenarios de prueba de principio a fin en el sistema.

Los cambios que se desarrollan en la herramienta ICOST se realizan periódicamente, aproximadamente una vez al mes, y cada cambio necesita ser probado, por lo que ahí es donde sería necesario enfocar las automatizaciones, dado que actualmente no se hacen pruebas exhaustivas, ni automatización de los procesos importantes como mínimo, esto es una de las razones por las cuales este proyecto viene a brindar soporte al proyecto ICOST y a Intel.

Es indispensable ejecutar las pruebas cada mes para poder asegurar la entrega de productos de calidad y de esta forma evitar incidentes con las finanzas e inventario de la empresa, cada prueba debe ser ejecutada, como se mencionó anteriormente, otra problemática identificada es que existen diversas pantallas, entre pantallas regulares y reportes por lo que no siempre se van a poder reutilizar datos pues son diferentes entradas de datos y tablas y hacer la revisión de forma manual conlleva tiempo.

Este proyecto propone la implementación de un marco de trabajo que permita el manejo de una herramienta que brinde al proyecto en sí un apoyo para brindar productos de calidad que cumplan con las expectativas sin necesidad de recurrir a reportes de problemas cuando ya estén los cambios y vayan a producción.

3. Justificación

Como solución a esta problemática como primer punto se propone la elección de una herramienta de automatización que sea viable, que permita reconocer todas las funcionalidades de la aplicación ICOST, mejor manejo de los objetos, que se adapte mejor a la organización, mayor facilidad de uso para los encargados de automatizar pruebas y analistas, que pueda automatizar, que pueda ejecutar consultas de bases de datos en SQL (por sus siglas en inglés Structured Query Language; en español lenguaje de consulta estructurada), que reconozca Excel, administración de tareas de Windows, convertir scripts en ejecutables de Windows (.exe).

Esta selección de la herramienta debe cumplir con todos los requisitos mencionados anteriormente para poder ser capaces de interactuar con la herramienta ICOST y poder cumplir con todas las necesidades de pruebas de software requeridas del mismo con esto nos aseguramos la máxima cobertura de las funcionalidades necesarias y de mayor impacto para el departamento.

Para la elección de las herramientas contamos con la facilidad de que en la misma empresa en la que realizamos el proyecto (Intel Costa Rica), cuenta con diferentes softwares de automatización, tanto libres como privativos, además de que existen expertos en el área de automatización que nos pueden brindar ideas y sugerencias de cómo afrontar la problemática actual.

Una vez seleccionada la herramienta de automatización, se propone como parte de la solución al problema un diseño e implementación de un marco de trabajo, en el cual tenga como base dos conceptos principales: enfoque basado en datos (en inglés data-driven approach) y enfoque basado en palabras clave (key-driven approach), los cuales nos van a permitir trabajar de manera ágil y eficaz ya que nos da la capacidad de automatizar múltiples casos de prueba con características similares pero de múltiples pantallas o reportes y ser manejados por medio de datos. Esto además nos permite tener una centralización de las automatizaciones lo que ha futuro permite una mayor facilidad a la hora de realizar mantenimiento de las pruebas automatizadas.

Después de analizar en conjunto la herramienta elegida y las características necesarias para desarrollo e implementación del marco de trabajo en el proyecto se continuará con la implementación de un plan piloto, el cual nos brindará un soporte para organizar las ideas de cómo enfocar el proyecto, de forma que sea introducido de la mejor manera, que se enfoque

correctamente en la problemática a solucionar y que no desarrolle ninguna ineficiencia o problemas a corto ni largo plazo.

Para comprobar que el nuevo marco de trabajo funcione correctamente, se realizará la ejecución de pruebas automatizadas de software basadas en el nuevo marco y en la herramienta elegida para esta implementación, se realizarán las pruebas necesarias con el fin de abarcar la mayoría de los casos que se podrían presentar durante una automatización del software ICOST, además el tener una base de cómo funciona la herramienta y los pasos que se realizaron para automatizar dichas partes será de mucha ayuda para los ingenieros o personal a cargo de la ejecución de las automatizaciones dentro del equipo de trabajo.

Luego de la ejecución de las pruebas automatizadas se realizará una evaluación de los resultados, corroborando si las pruebas obtuvieron el resultado esperado y si se cumplió a cabalidad las metas que se pretendían alcanzar por medio de la implementación del marco de trabajo, y en caso de resultar de forma satisfactoria se proseguirá con la implementación total del proyecto dentro del equipo ICOST después de mostrar los resultados obtenidos y contar con el aval del patrocinador.

4. Objetivos del Proyecto

4.1 Objetivo general

Implementar una propuesta de automatización de pruebas que permita el aseguramiento de la calidad de la herramienta ICOST mediante el desarrollo de un marco de trabajo.

4.2 Objetivos específicos

A. Investigar sobre marcos de trabajo de automatización de pruebas utilizados en proyectos similares para el aseguramiento de la calidad del software.
B. Analizar la forma de trabajo actual con la que se prueba la herramienta Evolve ICOST con el fin de comparar con casos exitosos de automatización de pruebas (los resultados de la investigación previa).
C. Diseñar un marco de trabajo para la automatización de pruebas que se ajuste a las necesidades del software Evolve ICOST.
D. Implementar un plan piloto basado en el diseño previamente definido para la ejecución automatizada de pruebas para el aseguramiento de la calidad del software.
E. Evaluar el impacto del nuevo proceso de automatización de pruebas mediante métricas con el fin de determinar el porcentaje de éxito para automatizar las pruebas en la herramienta Evolve ICOST.

CAPÍTULO II
MARCO TEÓRICO

CAPÍTULO II: MARCO TEÓRICO

Automatización

La automatización de procesos ha sido un sistema que ha venido en auge, debido a las grandes ventajas competitivas y administrativas que les brinda a las empresas en forma general, tomando como principal beneficio el tiempo invertido en los procesos, el cual siempre ha sido un factor muy importante en los negocios. La automatización es como su nombre lo dice automatizar procesos, hacerlos de forma automática, sin incurrir en retrasos por parte del factor humano; es la capacidad de las máquinas de realizar diferentes tareas de forma correcta sin ninguna intervención humana.

Como se menciona en el libro de Graham "Automating software testing can significantly reduce the effort required for adequate testing, or significantly increase the testing which can be done in limited time. Tests can be run in minutes that would take hours to run manually." [Automatizar pruebas de software puede reducir significativamente el esfuerzo requerido para las pruebas, o incrementar significativamente la cantidad de pruebas que se pueden realizar en un tiempo limitado. Las pruebas pueden ser corridas en minutos que manualmente llevarían horas.] (FEWSTER, M., & GRAHAM, D, 1999, p 3). El tiempo de diferencia entre automatizado y manual puede ser muy grande, hasta incluso de días.

Muchos procesos que se realizan en las empresas son muy repetitivos día a día, ahí es donde puede entrar a participar el proceso de automatización dentro de la empresa, pues al ser un proceso que no va a cambiar o que necesita ser realizado de la misma manera muchas veces fácilmente puede ser automatizado y de esta manera el tiempo que se invertía en esto puede ser utilizado en otro punto de mayor impacto.

Esto ha sido utilizado en diferentes áreas empresariales, ya sea de forma industrial (robots, máquinas, etc.) o en el área de informática (software), el impacto que logró tener dentro de las

empresas en general fue grande dado que afectó varias áreas en las que era necesaria, más en el punto que nos vamos a enfocar es en la automatización de procesos de software.

Las ventajas de la automatización son varias, como se mencionó anteriormente, el ahorro del tiempo es de alto impacto en una empresa, reducción de costos, acumulación de tareas, controlar los resultados, estandarizar las operaciones, minimizar la cantidad de errores provocados por el error humano, etc. Son muchos los beneficios de automatizar, aunque siempre existen algunas fallas que puedan evitar una automatización correcta, como lo sería el que falle el software de automatización, que se haya intentado automatizar un proceso que en realidad no puede ser automatizado completamente, pero terminan siendo muy puntuales. Algunos alegan que este proceso vino a quitar empleos, pues realiza el trabajo de las personas con mayor eficiencia, mientras que otros dicen que vino a proponer más empleo o al menos no a quitar empleos sino a cambiarlos, pues las máquinas y software necesitan mantenimiento y correcciones, por lo que se ocupan personas que realicen ese trabajo.

Un punto muy importante a la hora de automatizar es saber que procesos realmente deben o serían un beneficio al ser automatizados, el automatizar cualquier proceso existente no siempre es ganancia, algunos procesos pueden ser muy complejos de automatizar y debe ser revisado la relación entre el tiempo/dinero que lleva automatizarlo y el tiempo/dinero ahorrado al tenerlo automatizado. ya que esto puede conllevar a una pérdida importante. Por lo que la decisión de automatizar debe ser bien analizada por la organización realizando un análisis del proyecto, como se define en el siguiente texto “Automating tests is also a skill but a very different skill from testing. Many organizations are surprised to find that it is more expensive to automate a test than to perform it once manually. In order to gain benefits from test automation, the tests to be automated need to be carefully selected and implemented.” [Automatizar pruebas es también una habilidad, pero una habilidad muy diferente a probar. Muchas organizaciones se sorprenden de que sea más caro el automatizar una prueba que realizarla manualmente. Para obtener beneficios de la automatización de pruebas, las pruebas que serán automatizadas necesitan ser cuidadosamente seleccionadas e implementadas.] (FEWSTER, M., & GRAHAM, D, 1999, p 5)

Herramientas de automatización de software

La automatización de procesos es un proceso realizado por una herramienta que fue hecha específicamente para ese propósito, que dependiendo de lo que se quiera automatizar la herramienta puede variar, pues no todas las herramientas de automatización satisfacen las necesidades específicas del cliente. Las herramientas de automatización son variadas dentro de su campo, ya que dependiendo de la necesidad es la que elegimos, por ejemplo, para la elección de la herramienta se debe tomar en cuenta si la herramienta permite realizar scripts o es sólo por medio de identificadores, además, en caso de seleccionar la herramienta en la que se pueden realizar scripts se debe tomar en cuenta los lenguajes de programación que permite la herramienta.

Otra razón de la elección de la herramienta sería dependiendo de lo que se quiera automatizar, como si se quiere automatizar una página web, aplicación de escritorio, móviles, o todas, por lo que se debe elegir correctamente para no tener problemas con la plataforma, también un punto de los más importantes es si la herramientas de automatización se encuentra disponible para ser utilizada en el sistema operativo que se desea, además se debe tomar en cuenta en caso de que esta no sea de software libre hay que invertir dinero para su adquisición.

En fin, el uso de una herramienta de automatización nos brinda ayudas para mejorar la ejecución de nuestros procesos que normalmente llevarían una mayor cantidad de tiempo, pero no debemos olvidar que dependiendo de la necesidad se necesita una herramienta con ciertas características por lo que se debe tener cuidado y elegir correctamente, como se menciona “The way in which the tool is selected is very important in achieving this goal. If you choose the wrong tool, it may not do what you need or expect it to do. There may be significant technical difficulties in making the tool work in your environment” [El modo en que la herramienta es seleccionada es muy importante para alcanzar esta meta. Si se escoge la herramienta equivocada, puede no hacer lo que usted necesita que haga. Puede haber importantes dificultades técnicas haciendo que la herramienta trabaje en nuestro ambiente] (FEWSTER, M., & GRAHAM, D, 1999, p 249)

Cabe resaltar que en Intel se han realizado esfuerzos para automatizar en otros departamentos u otros proyectos, los cuales manejan distintas herramientas tales como Worksoft Certify, Parasoft SOATest, varios frameworks de Selenium y UiPath RPA. Podemos mencionar en el área de tecnologías de información utilizan Selenium para probar la página web de intel.com el cual es un proyecto sumamente grande, la ventaja que posee este proyecto es que cuentan con múltiples profesionales en el ámbito los cuales brindan soporte y desarrollo de automatizaciones. Otra área de TI es finanzas, los cuales prueban sus sistemas utilizando Certify, debido que Certify se ha establecido como la herramienta oficial para automatizar pruebas en SAP, como se sabe la mayoría de los grupos financieros utilizan SAP como base de sus funciones diarias, teniendo como ventaja el conocimiento de los procesos y además cuentan con un equipo especializado en soportar la herramienta Certify. Además, en Servicios Corporativos utilizan la herramienta SoaTest para realizar pruebas de API (Interfaz de programación de aplicaciones) las cuales les permite probar la comunicación entre las bases de datos y la interfaz.

Por otro lado, los departamentos fuera de IT, negocios como finanzas, cadena de suministros, recursos humanos, entre otros grupos, están optando por herramientas de RPA (Robotic Process Automation), las cuales permiten la automatización de procesos de negocio con herramientas de fácil uso y que personas no técnicas puedan desarrollar scripts, en Intel se optó por utilizar la herramienta UiPath RPA.

Aseguramiento de la calidad o Quality Assurance (QA)

Quality Assurance (QA) del inglés o Aseguramiento de la calidad es un proceso para determinar si el producto bajo prueba cumple los requerimientos especificados por el cliente. Al tener aseguramiento de calidad en proyectos ayuda a que la empresa entregue productos confiables y así mismo los clientes van a tener mayor credibilidad de la empresa
o equipo de desarrollo. También beneficia al mismo equipo de trabajo puesto que mejora los procesos de trabajo y mejora la eficiencia.

Se nos da como definición de calidad de software como “El conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas” ISO 8402 (UNE 66-001-92), con esto podemos ver que cumplir completamente con los requerimientos por parte del cliente es una parte vital para asegurar la calidad del software

Tipos de pruebas

En el área del aseguramiento de la calidad se realizan diversos tipos de testing para poder completar la tarea de entregar productos de calidad, cada tipo de testing se enfoca en un área diferente del producto, tanto testing de partes del producto como el producto entero y todas sus partes relacionadas.

Existen dos áreas grandes en los tipos de pruebas actualmente, entre ellas se encuentran las pruebas funcionales y las no funcionales, las pruebas funcionales se encargan específicamente de probar la funcionalidad del software más que todo que funcione como es esperado, por otro lado, las pruebas no funcionales como su nombre lo indica no se prueba la funcionalidad del software, sino que se enfoca en el comportamiento del mismo y la experiencia para con el usuario.
Se menciona que “It is impossible to perform all types of testing on a software as there is always fixed amount of time allocated for testing” [Es imposible el ejecutar todos los tipos de testing a un software ya que siempre hay una cantidad de tiempo fijado impuesto para pruebas](Hooda, I. H., & Singh Chhillar, R, 2015) , por ende se debe conocer los tipos más importantes de pruebas que existen y así saber cuáles se deben priorizar para tener un proyecto satisfactorio.

Con la idea de mostrar una parte importante del mundo del aseguramiento de la calidad, se pretende dar una explicación de los tipos más importantes actualmente tanto funcional como no funcional:

- Pruebas Unitarias (Unit Testing): Las pruebas unitarias consisten en pruebas a bajo nivel, básicamente probar a nivel de código fuente las funcionalidades de forma individual antes de incorporarlo al proyecto, es en sí probar cada pequeña funcionalidad del código por separado y evaluar que su resultado sea el esperado; cuando se realizan estas pruebas es bueno aislar la funcionalidad que está siendo probada de cualquier factor externo que pueda

influir en el resultado. La finalidad de estas pruebas es encontrar de forma anticipada un posible defecto que pueda costar más caro monetariamente al proyecto y arreglarlo con antelación antes de seguir avanzando.

- Pruebas de Componente (Component Testing): Las pruebas de componente son las pruebas que se enfocan en probar una funcionalidad general requerida por el negocio, es tomar pequeños módulos con una funcionalidad específica y ver que juntos cumplan una funcionalidad mayor, un ejemplo puede ser que una conexión a una base de datos se realizó correctamente.

- Pruebas de Integración (Integration Testing): Las pruebas de Integración consisten en probar la funcionalidad de varios componentes juntos, como por ejemplo que, si se inserta un registro a una base de datos, este se pueda ver reflejado en una pantalla.
- Pruebas de Regresión (Regression Testing): En estas pruebas se realizan cuando una nueva funcionalidad es agregada al proyecto, no siempre se sabe cómo puede afectar este cambio al resto de las funcionalidades del software por ende las pruebas de regresión consisten es volver a probar las funcionalidades ya existentes y comprobar que funcionen correctamente.

- Pruebas de Humo (Smoke Testing): Las pruebas de humo son las pruebas que se realizan a las funcionalidades principales y verificar que funcionen correctamente.

- Pruebas de aceptación (Acceptance Testing): Las pruebas de aceptación se enfocan en pruebas basadas en los requerimientos del negocio y se replica el comportamiento de los usuarios, con el fin de verificar que se cumpla con lo establecido en los criterios de aceptación.

- Pruebas de rendimiento (Performance Testing): Son pruebas que se encargan de medir tiempos de respuesta y el comportamiento de la aplicación bajo cierta carga de trabajo.

- Pruebas de estrés (Stress Testing): En esta prueba se expone el software a estrés de sobrecarga con el propósito de tener conocimiento de cuál es el límite y cómo reacciona el software a estas pruebas (ej: más usuarios de los que debería).

Marco de trabajo o Framework

Un marco de trabajo o framework (llamado en inglés) es una estructura creada para el desarrollo de una aplicación o proyecto en sí, los marcos de trabajo son estándares que permiten llevar el proyecto con un orden y una estructura que permita un fácil entendimiento y manejo en la forma que se va a utilizar la herramienta para desarrollar el proceso. “En otras palabras, un marco de trabajo o framework de desarrollo es una base desde donde se puede desarrollar algo más grande o más específico, se trata de una colección de código fuente, clases, funciones, técnicas y metodologías que faciliten el desarrollo de nuevo software.” (Minnetto, 2007)

Los marcos de trabajo que se vayan a utilizar dependen de lo que se quiera realizar, en caso de que sea el desarrollo de una aplicación pues habría que revisar el lenguaje en que se piensa desarrollar y el tamaño de la aplicación, pues existen muchos marcos de trabajo basados en un lenguaje que no van a servir para otros, en caso de no ser relacionado a desarrollo de código, los marcos de trabajo serían más específicos o más limitados ya que es más normal estandarizar para lenguajes de programación que para una herramienta que puede funcionar de diferentes formas dependiendo de lo que se quiera hacer.

La utilización de un marco de trabajo trae beneficios a corto y largo plazo, tanto para los que desarrollan en ese momento la aplicación como para los que revisen la aplicación tiempo después, viene a evitar que se repita el código o proceso que ya está implementado, agiliza el proceso de desarrollo de la aplicación o el mantenimiento de este, permite el uso de buenas prácticas ya preestablecidas, lo que agiliza la corrección de errores o su evasión.

“Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones.” (Gutiérrez, J, s.f.)

La creación de un marco de trabajo es un trabajo complicado pues se deben tomar en cuenta varios temas, como lo sería en caso de que ya exista uno con las funcionalidades que deseamos implementar, sería un gasto de tiempo, generar una buena estructura para la implementación y saber el enfoque que se le desea dar, por lo que hay que tener cuidado y estar claro de lo que se desea hacer.

Pruebas basadas en Datos o Data Driven Testing (DDT)

Data Driven Testing (DDT) del inglés o Pruebas basadas en Datos se basa en la creación de scripts de pruebas donde los datos o los valores de salida se leen de datos previamente definidos en vez de usar los mismos valores codificados cada vez que se ejecuta la prueba. “Consiste en la definición de scripts, donde se permiten definir variables. Las variables serán utilizadas para indicar los datos ingresados en los campos de la aplicación, permitiendo al script ejecutar la aplicación con datos proveídos externamente durante su ejecución” (Giménez M & Espínola, A, s.f)

De esta forma, los testers pueden probar cómo la aplicación maneja varias entradas de manera efectiva. Otra problemática que este tipo de pruebas llega a solventar es que cuando los datos de prueba deben actualizarse, el código de script debe ser cambiado para cumplir con las necesidades, en cambio al usar este método se debe actualizar la fuente de datos y no se debe cambiar el código fuente de los scripts. Otro problema de tener los datos de prueba dentro de los scripts es crear pruebas similares con datos diferentes. Esta ejecución puede ser fácil puesto que la persona solo debe copiar el script original y se cambia los datos. Pero al tener este tipo de reutilización no es tan efectivo a la hora de que se deba actualizar los scripts debido a que se debe cambiar la información en todos los scripts.

Al usar el enfoque de leer los datos de prueba desde una fuente externa del script y ejecutar la prueba en función a estos datos podemos evitar todos estos problemas mencionados ya que este enfoque nos facilita la edición de las pruebas por parte de personas sin habilidades técnicas. “Hoy en día, las pruebas basadas en datos (DDT) se vuelven muy importantes como parte del testing. En lugar de grabar múltiples pruebas para probar múltiples conjuntos de datos de entrada, es posible hacer que los scripts accedan a diferentes conjuntos de datos de entrada de tablas, u hojas de Excel, etc.” (Raj Kumari, M, 2011)

Pruebas basadas en Palabras Clave o Keyword Driven Testing

Keyword Driven Testing (KDT) del inglés o Pruebas basadas en Palabras Clave se basa en pruebas en existen palabras claves las cuales se leen para definir la prueba a ejecutar, en otras palabras, las palabras claves dirigen la automatización y las funcionalidades a ejecutar. “El marco de automatización basado en palabras clave se compone principalmente de tres partes: datos de prueba, bibliotecas de prueba y marco de automatización de prueba. Los datos de prueba contienen las entradas y salidas esperadas. La biblioteca de prueba es la interfaz del sistema bajo prueba y el marco. El marco de automatización de prueba lee los datos de prueba y maneja los errores cuando estas pruebas están en ejecución.” (Jain A & Sharma, S, 2012)

Este enfoque complementa de manera adecuada al enfoque antes visto (DDT), ya que al usar palabras claves como directivas para indicar qué funcionalidad ejecutar con los datos de prueba anteriormente mencionados en el enfoque de Prueba basado en datos. “Las pruebas basadas en palabras clave llevan el concepto incluso más allá al agregar palabras clave que conducen la ejecución de la prueba a los datos de la prueba.” (Laukkanen, P, 2006)

Control de versiones

Los sistemas de control de versiones son indispensables en el desarrollo de software sin embargo en muchas ocasiones también son requeridos en las pruebas automatizadas, dependiendo

de la herramienta, debido que nos brinda muchos beneficios, como lo mencionan Claudia, Diego y Edgar en sus investigación: “Los sistemas de control de versiones son aplicaciones que ayudan al proceso de desarrollo de software, facilitando la gestión del control de versiones de los archivos de código fuente generados por los desarrolladores, proporcionando herramientas para la fusión y generación de una nueva versión de un proyecto, permitiendo que múltiples desarrolladores trabajen en el mismo proyecto sin ocasionar pérdida de datos o bloqueos de archivos. Además, permiten recuperar archivos generados previamente, los cuales pueden ser utilizados para solucionar errores del sistema” (Tello-Leal, E, Sosa, C, Tello-Leal, D, 2012).

En Intel se cuenta con la herramienta GitLab como herramienta oficial para realizar control de versiones en proyectos, al utilizarla se asegura la integración de las pruebas automatizadas cuando múltiples desarrolladores están trabajando en el mismo proyecto, también la capacidad de recuperar archivos eliminados o perdidos, entre otros beneficios.

Ambientes de desarrollo

Los ambientes de desarrollo de software permiten el flujo correcto de desarrollo de una solución. Permite a su vez contar con un control más rígido sobre el proyecto y su estado. Ya que, si contamos con ambientes de desarrollo, pruebas, preproducción y producción qué son los establecidos en el proyecto ICOST y más utilizados en la industria se puede tener por separado cada función. En el ambiente de desarrollo es donde el desarrollador trabaja todo su código y puede realizar pruebas iniciales, en el ambiente de pruebas se realiza las pruebas para asegurarnos de la calidad del producto, el ambiente de preproducción el cual es una copia o respaldo de producción y por último el ambiente de producción es donde se encuentra la última versión del proyecto.

Niveles de pruebas

“Mike Cohn y Anand Bagmar, ubican a las pruebas unitarias y a las de aceptación como base en las pirámides de pruebas y las pruebas de interfaz gráfica (UI) se encuentran en el último

lugar" (Mascheroni, M, Mascheroni M & Irrazabal, E, 2016). Mike Cohn, autor del libro Succeeding with Agile, escribió una clasificación de cuáles pruebas se deben automatizar como prioridad, además nos brinda las ventajas de aplicar dicho método, es conocido como la pirámide de las pruebas. Representada con la siguiente ilustración:

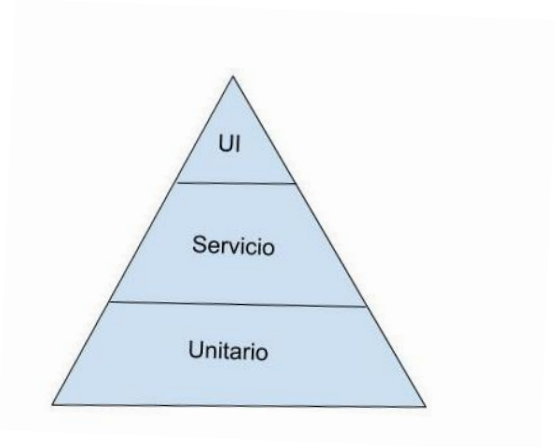


ILUSTRACIÓN 1 - PIRÁMIDE DE PRUEBAS. FUENTE: ELABORACIÓN PROPIA

La pirámide o los niveles de pruebas consiste en poner más esfuerzo en la automatización de la base, la cual es la prueba unitaria, llamada también unit tests en inglés, ya que son pruebas que se enfocan en probar cada elemento individualmente de manera independiente, así conseguir resultados más rápido a más bajo nivel en el cual los desarrolladores puedan corregir sin incurrir en gastos muy altos. "Pruebas unitarias automáticas, porque un primer punto primordial para detectar fallos es a nivel de desarrollador. Si una funcionalidad en este punto falla, podrían fallar pruebas de los siguientes niveles: integración, API etc." (Díaz Boente, M, 2017)

Las pruebas de servicio con el siguiente nivel de prioridad, estas pruebas suelen ser a nivel de API, en donde deseamos probar los servicios, la comunicación entre sí, estas pruebas son estables puesto que los servicios no cambian constantemente, lo cual nos permite automatizar de manera estable y generar resultados. "Pruebas a nivel de API, integración de componentes, servicios, que son los más estables y candidatos a automatizar." (Díaz Boente, M, 2017)

El último nivel se basa en las pruebas de interfaz, está de última prioridad puesto que son pruebas lentas en ejecución, además suele ser inestable ya que los elementos de la interfaz pueden cambiar constantemente, suelen ser más costosos puesto que con pruebas más complejas y abarcan más

escenarios para cubrir también porque el intentar corregir un error a este nivel suele ser más costoso que si se hubiera encontrado a nivel del desarrollador. “Pruebas de interfaz de usuario automatizadas. Ya que estas pruebas son variables, lentas en su ejecución y con muchas dependencias con otros componentes.” (Díaz Boente, M, 2017)

Herramientas más utilizadas en el ámbito de la automatización de pruebas

“El uso de una herramienta no garantiza el éxito en las pruebas, dado que para obtener resultados útiles es necesario conocer cómo utilizar estas herramientas y cómo interpretar sus resultados” (Díaz, J., Banchoff Tzancoff, C., Rodríguez, A., & Soria, V. (2009), como vemos no por el hecho de usar la herramienta vamos a tener un resultado satisfactorio, debemos seleccionar cuidadosamente la herramienta que vamos a utilizar y tomar en cuenta las diversas pruebas que se ejecutarán.

Actualmente tenemos a disposición diferentes herramientas de automatización que nos brindan diversas funcionalidades dependiendo de nuestra necesidad, algunas son herramientas libres (open source) y otras de pago, a continuación, se mostrarán algunas de estas herramientas y su funcionalidad:

1- **Selenium**: Es utilizada para la automatización de aplicaciones web, para hacer uso de esta herramienta de la mejor manera, el tester o el encargado de las pruebas debe tener habilidades en programación, ya que la herramienta está basada en código, algunos de los lenguajes en los que se pueden crear scripts para la automatización son Java, Groovy, Python, C#, PHP.

2- **Katalon Studio**: Esta herramienta nos permite realizar pruebas de API, pruebas de aplicaciones web, móviles y aplicaciones de escritorio, a diferencia de Selenium esta herramienta no está limitada a aplicaciones web.

3- **UFT** One: Al igual que Katalon puede ser utilizada para diferentes tipos de pruebas (pruebas web, aplicaciones de escritorio, móviles, API, y RPA (Robotic Process Automation))

4- **Test Complete**: Permite realizar pruebas para aplicaciones web, móviles, de escritorio, además se pueden crear scripts utilizando lenguajes de programación tales como JavaScript, VBScript, Python o C++.

5- **Postman**: Esta herramienta se enfoca en las pruebas de API, es una de las mejores herramientas para estas pruebas además de que es fácil de utilizar y amigable con el usuario.

6- **SoapUI**: Otra herramienta enfocada en pruebas de API, su versión de pago viene con herramientas avanzadas para pruebas, funciona para pruebas REST y SOAP.

Empresas en Costa Rica aplicando herramientas de automatización de pruebas

A continuación, se dará una pequeña muestra de información recopilada por medio de entrevistas con diferentes profesionales de varias empresas, las cuales están aplicando la automatización de pruebas con diversas herramientas.

Empresa	**Contacto**	**Herramienta o tecnología**	**Información**
Edify	David Álvarez	Postman TestNG Java	Se desarrolló un Framework desde cero para realizar Automation de GUI (Graphic User Interface) el cual permite desarrollar test cases y verificaciones a nivel de interfaz de usuario y asegurar un nivel de calidad y comprobación de requerimientos. Se usaron tecnologías como: TestNG, Log4j En Java, con patrón de diseño POM.

			Usando herramientas de API Testing como, por ejemplo: Postman logró probar componentes expuestos por Rest Services y así comprobar la funcionalidad y transaccional de módulos y componentes de la aplicación.
Mobilize	María Rodríguez	Selenium	Se desarrolló un framework desde cero para automatizar el UI de las aplicaciones web que se utilizan en la empresa. Crearon su propio framework basado en Selenium.
Gorilla Logic	David Álvarez	Selenium C# Cucumber	API Automation Testing Framework de Automation desarrollo en .net core 3.0, usando C#, RestSharp. Se prueba el API a través de endpoints para asegurar la respuesta de métodos y módulos del BE. GUI Automation Testing Framework de Automation desarrollo en .net core 3.0, usando C#, Page Object Model, Selenium, Cucumber. Se desarrollan pruebas de interfaz de usuario para verificar visibilidad de elementos visuales en las páginas web, así como transacciones comunes y flujos de usuario final.
Stryker	Mauricio Gamboa	Blue Prism UiPath	Se utilizan herramientas de RPA tales como Blue Prism y UiPath para automatizar los

			procesos de negocio. No hay mucho detalle puesto que están iniciando con el proceso.
Intel IT Intel.com	José Méndez	Selenium	En diversas áreas de Intel se automatiza con diversas herramientas, la decisión va a depender del tipo de automatización y del tipo de aplicación que se desea probar. En la página web de intel.com se utiliza Selenium como Framework de automatización, se desarrolló un Framework exclusivo para dicho software.
Intel Finanzas	Silvia Alpizar	UiPath	En el negocio se busca automatizar procesos en lugar de pruebas ya que se desea que los empleados puedan enfocar sus esfuerzos en tareas de más valor y no en tareas monótonas o repetitivas. Es por esto que han optado por tecnologías de RPA o Robotic Process Automation con la herramienta UiPath.
Intel IT Finanzas	Jean Pierre Araya	Certify	En el departamento de IT que brinda soporte a finanzas, utilizan el software SAP y además algunas páginas web, la herramienta que IT definió como herramienta de automatización oficial para SAP es Certify puesto que es estable y robusta. Brinda todas las funcionalidades requeridas. Todas las pruebas de UI las realizan con Certify en ambientes de pruebas.

TABLA 1 - EMPRESAS DE COSTA RICA APLICANDO TÉCNICAS DE AUTOMATIZACIÓN DE PRUEBAS. FUENTE: ELABORACIÓN PROPIA

CAPÍTULO III
METODOLOGÍA

CAPÍTULO III: METODOLOGÍA

1. Tipo de investigación

El enfoque de esta investigación es mixto, debido a que es necesario recabar pruebas cuantitativas donde generan datos como los tiempos de ejecución, el ahorro al tener las pruebas automatizadas, la cantidad de módulos, pantallas o reportes del sistema, entre otras, y a su vez cualitativas, debido a que debemos analizar procesos de pruebas mediante técnicas inductivas.

Roberto Hernández (Sampieri, 2014) en su libro "Metodología de la investigación", define que los métodos cuantitativos son aquellos que utilizan la recolección de datos para probar hipótesis con base en la medición numérica y el análisis estadístico, con el fin establecer pautas de comportamiento y probar teorías.

Para recabar la información necesaria para llevar a cabo este proyecto, se utilizaron los instrumentos cualitativos, tales como la observación, entrevistas y grupos focales. Además, se utilizó el instrumento cuantitativo llamado encuesta. Estos instrumentos son descritos más adelante.

2. Población y muestra

La población para este proyecto serán todos los miembros del equipo de trabajo de ICOST e integrantes del grupo de automatización de Intel, se dividirán de acuerdo con sus roles, esto para tener una perspectiva más variada sobre cómo percibe el proyecto cada área de trabajo y tener un mayor entendimiento, además como muestra intentaremos utilizar la mayor cantidad de población posible, calculando el porcentaje de población utilizado con cada herramienta en la siguiente tabla.

Población	**Muestra según herramientas**

	Observación	Encuesta	Grupo Focal
Administradores del Proyecto (Project Managers PO)	NA	100% de población	100% de población
Analistas del Sistema (System Analysts SA)	25% de población	50% de población	50% de población
Facilitador de Scrum (Scrum Master SM)	NA	NA	50% de población
Desarrolladores (Developers Devs)	25%	50% de población	NA
Aseguramiento de la Calidad (Quality Assurance Analysts QA)	100% de población	100% de población	100% de población
Grupo de Automatización	20% de población	NA	20% de población
Sistema ICOST	50% del sistema	NA	NA

TABLA 2 - POBLACIÓN Y MUESTRA

3. Descripción de instrumentos

Observación:

La observación consiste en realizar como su nombre lo dice, es una observación del problema, tomar datos y analizarlos, suele requerir bastante tiempo, pero nos permite tener una perspectiva amplia de lo que se quiere lograr.

Roberto Hernández Sampieri menciona “En la investigación cualitativa necesitamos estar entrenados para observar, que es diferente de ver (lo cual hacemos cotidianamente). Es una cuestión de grado. Y la ‘observación investigativa’ no se limita al sentido de la vista, sino a todos los sentidos” (Sampieri, 2014), lo que nos da a entender es que se necesita tomar gran atención a los detalles.

Usar un método nos permite recolectar datos reales ya que podemos observar de primera mano cómo se realizan los procesos en ICOST en nuestro caso, esto nos brinda la facilidad de ver que sucede durante el proceso de pruebas ejecutado, como reacciona la herramienta ICOST durante este proceso.

Por medio del análisis del trabajo de diferentes miembros del equipo de trabajo se genera la observación, con el propósito de realizar una recolección de datos amplia, dependiendo del miembro del equipo que estemos observando se pueden ver que diferentes variables afectan el proceso de pruebas, que procesos toman más tiempo realizar y así evitar verlo desde una sola perspectiva; la recolección de datos se planea realizar por medio del método “shadowing” y reuniones.

La recolección de datos se trabaja por medio de una plantilla que se encontrará en los anexos, los datos que serán recolectados se basarán en los que se consideren de mayor relevancia para mejorar el proceso de pruebas de ICOST y para completar el proyecto de mejor manera, no toda la información necesaria será recolectada de este modo, necesitaremos hacer uso de otros instrumentos de recolección de información.

Encuesta

La encuesta es un método tradicional en las investigaciones cuantitativas que permite recopilar información sistemática capaz de dar respuestas a problemas tanto en términos descriptivos como en relación con las variables definidas en el apartado 3.6. Para poder asegurar que la información va a servir, es necesario establecer un diseño que asegure el rigor de la información recolectada.

Normalmente a través de las encuestas se pueden encontrar descripciones de los objetos de estudio, patrones y relaciones entre las características investigadas. Las encuestas pueden servir de instrumento exploratorio para identificar variables y relaciones y medir variables de la investigación.

Algunas ventajas de las encuestas son especiales para recopilar opiniones, creencias o actitudes, además esta técnica es utilizada cuando no se puede utilizar la técnica de la observación directa por factores económicos o contextuales. Y se utilizan cuando se quiere generalizar el resultado a una población definida. (Rodríguez, 2010)

Algunas de las desventajas de esta técnica es que es difícil establecer relaciones causales y además no toma en cuenta los factores contextuales que pueden interferir en las respuestas del sujeto. (Rodríguez, 2010)

Las encuestas realizadas a desarrolladores, dueños del producto y analistas de sistemas de ICOST, se desarrollan en la plataforma digital de Google llamada “Google Forms”. Estas encuestas van a ser difundidas por medio del correo electrónico interno de la empresa. Además, dicha plataforma nos permite almacenar los datos finales, los cuales se recopilaron en el anexo 27.

Entrevistas

Se nos define entrevista como “La entrevista es la técnica con la cual el investigador pretende obtener información de una forma oral y personalizada. La información versará en torno a acontecimientos vividos y aspectos subjetivos de la persona tales como creencias, actitudes, opiniones o valores en relación con la situación que se está estudiando.” Torrecilla, J. M. (2006). La entrevista. Madrid, España: Universidad Autónoma de Madrid.

Como se puede observar la entrevista es una forma muy directa de obtener la información necesaria de la fuente, los datos recolectados por medio de la entrevista pueden ser cuantitativos o cualitativos.

Los datos cuantitativos serían los que se pueden medir y obtener datos numéricos que se utilizan para la creación de estadísticas para su análisis por otro lado tenemos los datos cualitativos, que serían los datos que no se pueden medir, se recolectan datos, pero en su mayoría basados en la opinión del entrevistado, lo cual nos permite tener una entrevista más flexible y abierta a discusión.

En el caso del proyecto se realizarán entrevistas de carácter cualitativos, pues el enfoque es el cómo se puede solucionar el problema actual de ICOST con respecto a las pruebas manuales, por lo que es importante conocer las opiniones generales y objetivas de los diferentes entrevistados, mientras que no son tan importantes lo números estadísticos sobre el tema.

Se planifican reuniones con diferentes miembros de la población para obtener su perspectiva del proyecto, algunos de los enfoques que se darán en estas entrevistas son el cómo mejorar el proceso de pruebas, cuales fallas piensan que hay actualmente, en qué afecta que no se realicen todas las pruebas necesarias, duración actual de las pruebas y el por qué.

A continuación, en la siguiente tabla describimos las entrevistas realizadas para recabar información relevante del proyecto:

Fecha	Participantes	Descripción	Anexo
05 noviembre, 2018 3:00 PM- 4:00 PM	Scott McDonald, Juan Diego González y Mariela Sequeira	Entrevista realizada al líder de calidad del proyecto ICOST para recopilar información específicamente sobre las pruebas de interfaz realizadas manualmente.	Minuta de reunión #1
12 diciembre, 2018 2:00 PM- 2:30 PM	Nazia Khan, Juan Diego González y Mariela Sequeira	Entrevista a líder de un equipo de IT que se especializa en determinar las mejores prácticas de Calidad y brindan las herramientas necesarias para automatización de pruebas. El fin de dicha entrevista fue alinear la propuesta solución con las mejores prácticas establecidas para IT.	Minuta de reunión #2
21 enero, 2019 10:00 AM - 10:30AM	Jacqueline Miranda, Juan Diego González y Mariela Sequeira	Entrevista realizada a dueña del producto y patrocinadora de este proyecto con el fin de entender los puntos más críticos en el proceso de pruebas del sistema ICOST.	Minuta de reunión #3
19 febrero, 2019 09:00 AM - 09:30 AM	Levi Chang, Juan Diego González y Mariela Sequeira	Se le realizó una entrevista al dueño del producto principal, con el fin de entender más a fondo la problemática y además exponer la posible solución.	Minuta de reunión #4
08 marzo, 2019 11:00 AM - 12 MD	John Sullivan, Juan Diego González y Mariela Sequeira	Entrevista realizada a experto de la herramienta Certify con el fin de obtener guía y aprobación, además para obtener accesos.	Minuta de reunión #5
12 abril, 2019 02:00 PM- 02:30 PM	Juan Carlos Murillo, Juan Diego González y Mariela Sequeira	Se realizó una entrevista al facilitador del proyecto con el fin de recopilar información desde su perspectiva, además para conversar sobre la posible solución.	Minuta de reunión #6

06 junio, 2019 11:00 AM - 11:30 AM	Michael Weir, Juan Diego González y Mariela Sequeira	Se le realizó una entrevista al jefe de producto, con el fin de obtener retroalimentación de la solución expuesta.	Minuta de reunión #7
07 agosto, 2019 04:00 PM - 05:00 PM	Jorge Grimaldi, Juan Diego González y Mariela Sequeira	Entrevista a desarrollador de ICOST con el fin de entender los procesos de SQL más importantes y entender las funciones a probar.	Minuta de reunión #8
16 octubre, 2019 02:00 PM - 02:30 PM	Srikanth Bongu, Juan Diego González y Mariela Sequeira	Entrevista realizada a desarrollador del proyecto para entender desde el punto de vista de un desarrollador y entender posibles escenarios faltantes.	Minuta de reunión #9
11 noviembre, 2019 04:00 PM - 4:30 PM	Juan Carlos Murillo, Juan Diego González y Mariela Sequeira	Entrevista a facilitador del proyecto ICOST con el fin de obtener retroalimentación del trabajo realizado al momento.	Minuta de reunión #10
18 febrero, 2020 02:00 PM - 3:00 PM	Jesse Fitterer, Juan Diego González y Mariela Sequeira	Entrevista realizada con un experto de la herramienta SOATest para entender la herramienta, funcionalidades, limitaciones, entrenamientos, soporte, entre otros aspectos.	Minuta de reunión #11
07 mayo, 2020 04:00 PM - 4:30 PM	Jorge Grimaldi, Juan Diego González y Mariela Sequeira	Entrevista a desarrollador del proyecto ICOST con el fin de obtener retroalimentación del trabajo realizado.	Minuta de reunión #12
23 junio, 2020 02:00 PM - 3:00 PM	Soledad Riggioni, Juan Diego González y Mariela Sequeira	Entrevista a dueña del producto ICOST con el fin de obtener retroalimentación del trabajo realizado.	Minuta de reunión #13

TABLA 3 - ENTREVISTAS APLICADAS A EMPLEADOS DE INTEL. FUENTE: ELABORACIÓN PROPIA

Las minutas de estas se pueden visualizar en los anexos del 2 al 14.

Grupo focal

Los grupos focales según lo define Zavaleta son un método efectivo de recolección de información cualitativa en el cual uno o dos investigadores y varios participantes se reúnen en una sesión grupal para discutir en torno a un tema específico que generalmente gira en torno al objetivo de la investigación. (Zavaleta, Y, 2013)

Saldanha y compañeros definen el grupo focal como un proceso dinámico en el que los participantes intercambian ideas, de forma que sus opiniones pueden ser confirmadas o contestadas por otros participantes. La técnica del grupo focal no busca consensos, de modo que los participantes pueden mantener las opiniones iniciales, cambiarlas o adoptar nuevas ideas. (Saldanha , D, Colomé, C, Heck, T, Nunes, M, Viero, V, 2015)

Algunas ventajas de los grupos focales son la posibilidad de fomentar el acceso a información acerca de un proceso o fenómeno, ya que la interacción entre los participantes brinda los diversos puntos de vista y opiniones. Además, beneficia a los investigadores ya que permite el ejercicio crítico al promover una discusión abierta sobre temas específicos. Los grupos focales proporcionan el incentivo a respuestas significativas o ideas nuevas, al mismo tiempo que instiga opiniones contrarias. (Saldanha, D, 2015)

Los grupos focales son enfocados a dueños del producto, facilitador del proyecto, analistas de sistemas, líder de calidad, y desarrolladores de ICOST, se desarrollan en la plataforma digital de Microsoft llamada “Skype for Business” y recientemente hemos incorporado “Microsoft Teams”, además de reuniones presenciales en salas dentro de las oficinas de Intel Costa Rica.

A continuación, en la siguiente tabla describimos los grupos focales realizados a lo largo del proyecto:

Fecha	Participantes	Descripción	Anexo
07 enero, 2019 09:00 AM - 09:30 AM	Scott McDonald Levi Chang Juan Diego González Mariela Sequeira	Se realizó una reunión con el líder de calidad y el jefe del producto, para obtener retroalimentación de la solución propuesta específicamente en Certify para las pruebas de regresión con el fin de obtener críticas constructivas.	Minuta de reunión #14
07 marzo, 2019 01:00 PM - 01:30 PM	Andrés Rodríguez Joseph Campos Heizel Perez Juan Diego González Mariela Sequeira	Reunión con diversos ingenieros de automatización de diversos proyectos dentro de IT, con el fin de conocer más opciones y herramientas y con el fin de obtener nuevas ideas.	Minuta de reunión #15
10 mayo, 2019 04:00 PM - 04:30 PM	Soledad Riggioni Adrián Campos Zaily Rojas Juan Diego González Mariela Sequeira	Reunión con analistas del sistema ICOST con el fin de revisar los escenarios de pruebas.	Minuta de reunión #16
26 agosto, 2019 03:30 PM - 04:00 PM	Juan Carlos Murillo Juan Diego González Mariela Sequeira	Se realizó una reunión con el líder para obtener retroalimentación de la solución propuesta además de mostrar el nuevo proceso automatizado para las pruebas de regresión en específico con el fin de obtener críticas constructivas.	Minuta de reunión #17
17 octubre, 2019 11:30 AM- 12:00 PM	Jacqueline Miranda - Dueña del producto Jean Marc Lenc - Líder Carlos Ovares - Analista de sistema Mauricio Otarola - Jefe Juan Diego González	Se realizó una reunión con líderes y personas clave para conversar sobre el proceso de regresión de pruebas en el proyecto ICOST y cómo la automatización beneficia dicho proceso.	Minuta de reunión #18

	Mariela Sequeira		
28 octubre, 2019 02:00 PM - 03:00 PM	Jorge Grimaldi Daniela Castro Eduardo Rojas Katherine Corrales Juan Diego González Mariela Sequeira	Se obtuvo retroalimentación de parte de los desarrolladores específicamente de la solución desarrollada en la herramienta SOATest	Minuta de reunión #19
14 enero, 2020 2:00 PM- 3:00 PM	Jesse Fitterer - Experto en SOATest John Sullivan - Experto en Worksoft Certify Juan Diego González Mariela Sequeira	Reunión realizada con el fin de reunir expertos de ambas herramientas utilizadas en el proyecto, con el fin de ver la solución que se estaba desarrollando tanto como en Certify como en SOATest, además para obtener retroalimentación, guía, ideas nuevas, generar crítica constructiva y consejos que pudiéramos aplicar con las tecnologías que poseemos.	Minuta de reunión #20
30 enero, 2020 4:00 PM - 4:30 PM	Juan Carlos Murillo Juan Diego González Mariela Sequeira	Se hizo un demo al facilitador del proyecto, sobre la automatización utilizando las herramientas para recopilar retroalimentación.	Minuta de reunión #21
19 febrero, 2020 2:00 PM - 3:00 PM	Jorge Grimaldi Daniela Castro Eduardo Rojas Katherine Corrales Juan Diego González Mariela Sequeira	Se desarrolló el tema de Data-driven específicamente para la herramienta SOATest con los desarrolladores de ICOST.	Minuta de reunión #22
01 abril, 2020 3:00 PM - 4:00 PM	Jorge Grimaldi Daniela Castro Eduardo Rojas Katherine Corrales Juan Diego González Mariela Sequeira	Se obtuvo retroalimentación de parte de los desarrolladores sobre la solución implementada.	Minuta de reunión #23

TABLA 4 - GRUPOS FOCALES APLICADOS A EMPLEADOS DE INTEL. FUENTE: ELABORACIÓN PROPIA

Las minutas de estas se pueden visualizar en los anexos del 15 al 24.

4. Procedimientos para analizar la información del diagnóstico

Con el fin de analizar la información se documentará cada una de las entrevista y grupos focales en forma de minutas para posteriormente por medio de la lectura de los resultados analizar y generar una solución acorde a la problemática real. Al utilizar todos los métodos anteriormente descritos se podrá obtener datos reales y no apreciaciones subjetivas. Además, se realizará una comparación de las respuestas de las distintas personas involucradas para lograr realizar un consenso de los puntos de enfoque para una solución óptima, encontrar temas recurrentes y facilitar la toma de decisiones sobre los distintos aspectos de la solución.

CAPÍTULO IV
PROPUESTA DE SOLUCIÓN

CAPÍTULO IV: PROPUESTA DE SOLUCIÓN

1. Diagnóstico

Tras analizar la información recopilada por la observación, entrevistas y grupos focales, a continuación, se va a detallar especificaciones para una solución apta para el proyecto Icost, así como las principales funcionalidades a cubrir en las pruebas, información relevante sobre posibles beneficios a obtener gracias a la información actual y manual de los procesos.

Se logró analizar las principales funcionalidades a probar en cuanto a las pruebas de UI, algunas de ellas son: validación de acceso a pantallas con distintos roles, validación de los filtros por pantalla o reportes, validación de los resultados generados por la pantalla o reporte como lo es la validación de las columnas correctas en la tabla de resultados, validación de un reporte histórico, entre otras. Asimismo, de priorizar una muestra de pantallas y reportes para el desarrollo del proyecto.

Gracias a la observación hemos analizado un punto de preocupación de parte de los líderes y del equipo de trabajo, el cual es el aseguramiento de la calidad en los cambios realizados a nivel de código en SQL, es por esto que hemos incorporado pruebas de enfoque unitario para probar procedimientos almacenados, así mismo hemos identificado los principales procedimientos almacenados a probar en la solución.

Con la información analizada hemos obtenido una serie de expectativas del dueño del producto, así como un estimado en tiempos manuales a la hora de ejecutar las pruebas de regresión y unitarias, y su frecuencia de ejecución.

2. Propuesta de solución

La solución es propuesta a partir del diagnóstico de la situación actual y se basa en una propuesta de marco de trabajo para las pruebas automatizadas, generada del conocimiento adquirido a partir del marco teórico, entrenamientos en el ámbito y conocimientos propios.

Con este modelo se busca definir la forma de automatizar las pruebas tanto de interfaz como unitarias para que sea de manera efectiva y de fácil manejo por los colaboradores de ICOST.

La solución busca cubrir mayor parte de la pirámide de Cohn como ha sido explicada en el marco teórico, a partir de las entrevistas y grupos focales se ha determinado nuestro foco de atención en la capa de las pruebas a nivel de interfaz gráfica y en la capa de las pruebas unitarias.

Así mismo por medio de la observación logramos analizar que en cada proceso de pruebas ya sea de un cambio grande en la aplicación o pruebas de regresión, les toma tiempo y varios recursos en probar manualmente a nivel de la página web validar la funcionalidad de las pantallas y reportes, como también identificamos que en cada cambio de código no hay un proceso estándar para probar el nuevo código que está siendo añadido a la aplicación por lo que consideramos necesario este nivel de pruebas automatizadas.

La solución involucra dos niveles de pruebas por lo tanto se debe buscar ya sea una herramienta adecuada para ambos niveles o dos herramientas, una por nivel, en este caso Intel cuenta con licencias corporativas de varias herramientas de automatización, como lo hemos analizado anteriormente, la herramienta Worksoft Certify es la ideal para las pruebas de interfaz gráfica ya que nos facilita la automatización de páginas web y así mismo ofrece una interfaz amigable al usuario por lo que no se requiere una persona experta en desarrollo de software, además ofrece mecanismos para poder desarrollar varias técnicas que han sido analizadas en este proyecto como Data-driven y Keyword-drive. Para el nivel de pruebas unitarias, ya que la aplicación ICOST el backend está compuesto por procedimientos almacenados en SQL, se determinó que la herramienta Parasoft SOAtest es ideal para dicho objetivo y nos ofrece al igual que Certify mecanismos avanzados para un mejor desarrollo de pruebas automatizadas, así mismo con dicha herramienta se puede cubrir un nivel más de la pirámide como lo es el nivel de servicio para automatizar pruebas a nivel de API, no nos enfocaremos en este proyecto en estas pruebas puesto que en el proyecto ICOST no cuenta con gran volumen de

APIs a este momento.

Durante las entrevistas se determinó que las pruebas con API no se realizaban en este momento, pero al contrario de las pruebas de unidad con procedimientos almacenados, no se cuenta actualmente con una cantidad significativa de APIs ya desarrolladas o listas para pruebas, por lo que se optó el realizar un prototipo de pruebas, con las que se posteriormente se podrán guiar para la implementación final de las pruebas de API.

Según hemos identificado con la ayuda de miembros del equipo de ICOST las pantallas y reportes a nivel de interfaz gráfica con mayor volumen de usuarios y de mayor importancia. La lista se muestra a continuación:

Tipo	**Nombre**
Pantalla	Adjust Last Available Cost
Pantalla	PRQ Dates
Pantalla	Cost Center Mapping
Pantalla	Cost Per Unit for FG Procurement
Pantalla	LCM Adjustments
Pantalla	Demand and ASP
Pantalla	Direct Materials Piece Part Pieces
Pantalla	Div Eng Parameter
Pantalla	DPW Adjustments
Pantalla	Overhead Allocator Pools
Reporte SSRS	Finished Goods Inventory
Reporte SSRS	Scrap Report (Bonus)
Reporte SSRS	Scrap Report (FG)
Reporte SSRS	Scrap Report (WIP)
Reporte SSRS	WIP EOH Inventory by Company Code

Reporte SSRS	Sales by FIFO Layer
Reporte SSRS	Stage Outs
Reporte SSRS	AT Product Cost
Reporte SSRS	Costed Operation Details
Reporte SSRS	Finished Goods Movement

TABLA 5 - LISTA DE PANTALLAS Y REPORTES SSRS PRIORIZADOS PARA LAS PRUEBAS AUTOMATIZADAS Y MARCO DE TRABAJO. FUENTE: ELABORACIÓN PROPIA

De igual forma a partir de la observación y de entrevistas con los expertos en el área hemos identificado algunos procedimientos almacenados de mayor relevancia o procedimientos almacenados con cambios recientes para aprovechar dicha prueba automatizada. La lista se muestra a continuación:

Nombre del procedimiento almacenado	**Nombre de la base de datos**
usp_un_lsp_trns_stg_uom_flip	Staging
usp_cibr_catch_up_ur_archv_ins	Staging
usp_cibr_catch_up_cibr_trns_archv_ins	Staging
usp_cibr_catch_up_md_archv_ins	Staging
usp_md_pce_part_excl_get_grid	Staging
usp_get_fact_wafer_cost	Staging
usp_allct_fnl_wbs_spnd_snpsht	ICOST
usp_allct_actual_get_vol_cost_centr_map_cost_centr_dtl	Staging
usp_actual_dp_cost_center_map_Cost_center	Staging
usp_allct_actual_upd_cost_centr_map_data	Staging

TABLA 6 - LISTA DE PROCEDIMIENTOS ALMACENADOS PRIORIZADOS PARA LAS PRUEBAS AUTOMATIZADAS Y MARCO DE TRABAJO. FUENTE: ELABORACIÓN PROPIA

A continuación, se explica cada nivel de prueba, así como la herramienta elegida y su marco de trabajo:

Nivel de UI o interfaz gráfica - Worksoft Certify

Certify es una herramienta de automatización de pruebas a nivel de interfaz gráfica, permite a los usuarios crear pruebas a partir de mapas con los objetos de páginas web entre otras tecnologías, por lo tanto no involucra código, esto facilita su uso ya que cualquier usuario, independientemente si es desarrollador o no, puede crear pruebas automatizadas, por otro lado Intel ofrece los entrenamientos necesarios de manera gratuita a sus empleados y posee la licencia corporativa así cualquier empleado puede solicitar acceso a la herramienta.

Certify ofrece una serie de funcionalidades aptas para la implementación de métodos como Keyword-Driven y Data-driven los cuales pueden ser manejados por recordsets (o conjunto de registros en su traducción al español), estos son archivos que contienen datos utilizados en las pruebas, además permite a los procesos ingresar o verificar los datos.

El primer concepto de nuestro marco de trabajo es la versatilidad para elegir de manera fácil y ágil el ambiente de desarrollo en el cual se desea ejecutar las pruebas automatizadas. En este caso utilizamos recordsets para la definición de los ambientes de ICOST, cada uno contiene la información sobre el URL de la página web correspondiente. En la siguiente figura se muestran los recordsets, al lado izquierdo las definiciones de los ambientes tales como DEV, QA y Sandbox, y al lado derecho un ejemplo de los datos que contienen. Como se puede ver en la imagen la tarea de cambiar de ambiente depende de seleccionar de un dropdown el ambiente deseado.

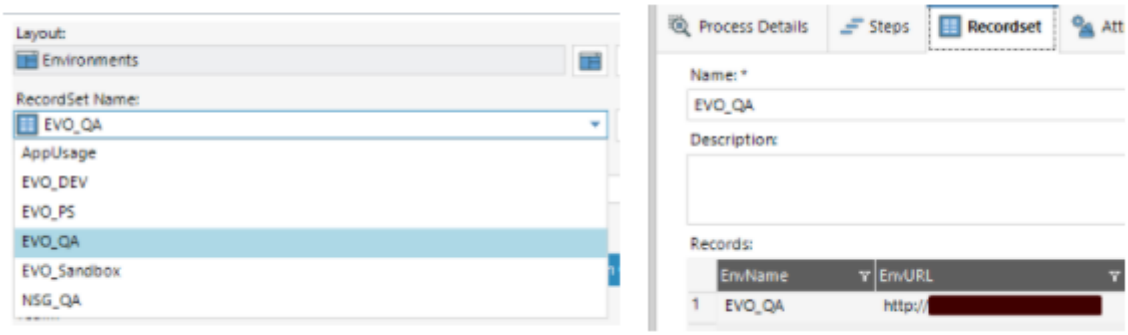
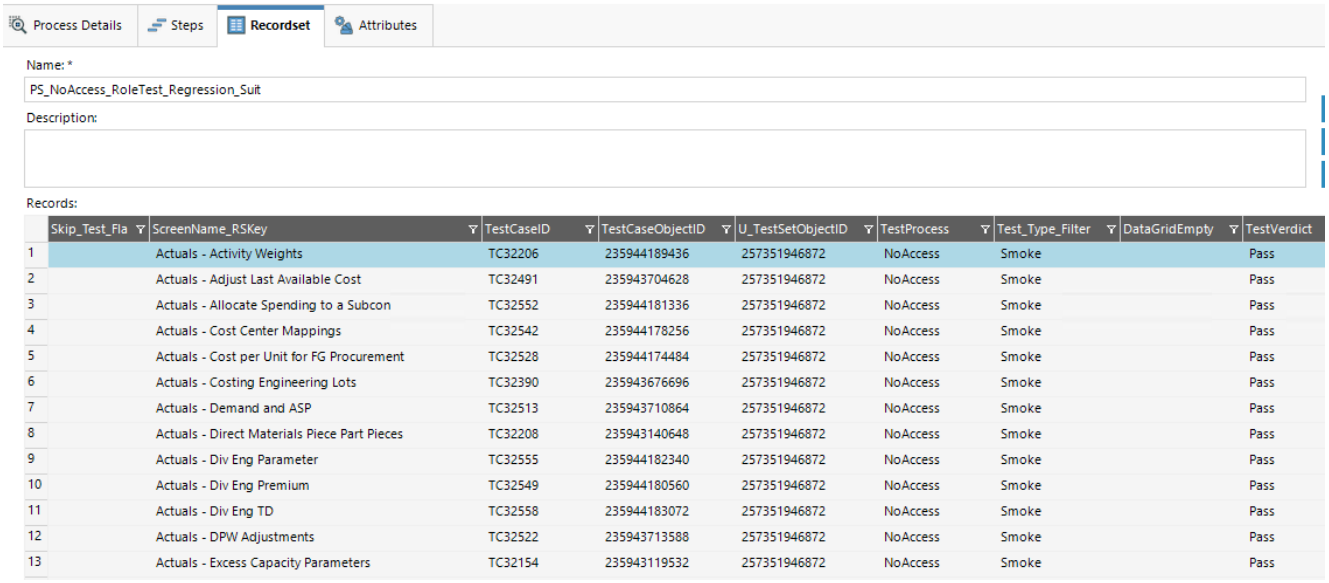


ILUSTRACIÓN 2 - RECORDSET DE AMBIENTES. ELABORACIÓN PROPIA.

Siguiendo este mismo concepto de versatilidad, el manejo de las pruebas se realizó utilizando varios recordsets para controlar la ejecución y el proceso a probar. El primero de ellos contiene la siguiente información básica de las pantallas o reportes:

- Bandera para saltar la prueba: esto nos ayuda a saltar pruebas que no deseamos ejecutar, al poner la letra Y, esto indica al proceso que queremos saltar dicha prueba.
- Nombre de la pantalla o reporte: compuesto por dos partes, el nombre del sistema - el nombre de la pantalla o el reporte. Como por ejemplo Actuals - Activity Weights
- ID del caso de prueba, ID de objeto del caso de prueba e ID de objeto del set de pruebas en Rally: la aplicación Rally es utilizada para el manejo de la documentación de las pruebas por IT en Intel, por lo tanto, el ID ayuda a incorporar trazabilidad con respecto a las automatizaciones y su documentación. Así como los ID de los objetos tanto del caso de prueba como del set de pruebas, ayudan a publicar el resultado de la automatización en Rally y llevar un mejor control visual del progreso de una fase de pruebas.
- Proceso o funcionalidad por ejecutar: en esta columna es donde se pone en práctica el uso de keyword-driven testing, ya que es acá donde se especifica cuál funcionalidad se quiere probar. Son palabras claves ya establecidas, descritas más adelante.
- El resultado: este dato es generado por la misma automatización con el fin de mostrar el resultado final de la prueba al final de la ejecución.



	Skip_Test_Fla	ScreenName_RSKey	TestCaseID	TestCaseObjectID	U_TestSetObjectID	TestProcess	Test_Type_Filter	DataGridEmpty	TestVerdict
1		Actuals - Activity Weights	TC32206	235944189436	257351946872	NoAccess	Smoke		Pass
2		Actuals - Adjust Last Available Cost	TC32491	235943704628	257351946872	NoAccess	Smoke		Pass
3		Actuals - Allocate Spending to a Subcon	TC32552	235944181336	257351946872	NoAccess	Smoke		Pass
4		Actuals - Cost Center Mappings	TC32542	235944178256	257351946872	NoAccess	Smoke		Pass
5		Actuals - Cost per Unit for FG Procurement	TC32528	235944174484	257351946872	NoAccess	Smoke		Pass
6		Actuals - Costing Engineering Lots	TC32390	235943676696	257351946872	NoAccess	Smoke		Pass
7		Actuals - Demand and ASP	TC32513	235943710864	257351946872	NoAccess	Smoke		Pass
8		Actuals - Direct Materials Piece Part Pieces	TC32208	235943140648	257351946872	NoAccess	Smoke		Pass
9		Actuals - Div Eng Parameter	TC32555	235944182340	257351946872	NoAccess	Smoke		Pass
10		Actuals - Div Eng Premium	TC32549	235944180560	257351946872	NoAccess	Smoke		Pass
11		Actuals - Div Eng TD	TC32558	235944183072	257351946872	NoAccess	Smoke		Pass
12		Actuals - DPW Adjustments	TC32522	235943713588	257351946872	NoAccess	Smoke		Pass
13		Actuals - Excess Capacity Parameters	TC32154	235943119532	257351946872	NoAccess	Smoke		Pass

ILUSTRACIÓN 3 - RECORDSET DE PANTALLAS PARA LA VALIDACIÓN DE SEGURIDAD. ELABORACIÓN PROPIA.

Con este recordset se busca establecer las pruebas de las diversas pantallas y reportes a probar, como se puede ver en la imagen anterior cada línea del recordset es una prueba o cubre una pantalla o reporte. Los usuarios deben llenar el recordset el cual es sencillo y con esto realizó una prueba automatizada sin haber realizado un solo script en Certify. Este es el gran beneficio de Data-Driven Testing, además de que el mantenimiento es más ágil puesto que solamente se debe verificar los recordsets y no muchos scripts por separado lo que implicaría una complejidad más elevada. La columna de Proceso a ejecutar (TestProcess) nos indica cual funcionalidad se quiere probar esto ayuda a enfocar las pruebas y poder personalizar las ejecuciones ya que no siempre se desea ejecutar todas las pruebas de regresión y se está aplicando el concepto KeyWord-driven debido que con palabras definidas como NoAccess, ValidAccess, ReadOnlyAccess, ScreenUI, entre otras, dirigimos las pruebas a ejecutar el proceso en específico, esto lo explicaremos más a detalladamente más adelante.

El siguiente recordset contiene información más específica de cada pantalla o reporte como los datos a probar, los filtros y botones visibles, entre otros. Esto se lleva a cabo con la creación de subprocesos genéricos con funcionalidades específicas y recibiendo parámetros por medio de los recordsets. Los datos de este recordset se explican a continuación:

- Bandera para ingresar datos en los filtros: este dato es utilizado para indicarle al proceso si debe seleccionar filtros en la pantalla o reporte, se utiliza la letra Y. De lo contrario no realiza la acción de selección de filtros.
- Nombre del filtro: nombre visible del filtro a seleccionar, con esto se valida la existencia del filtro.
- Bandera para validar los datos del filtro
- Selección del filtro: este dato indica qué tipo de filtro es, entre las opciones están selección única, selección múltiple o combo boxes.
- Valor del filtro: el dato a seleccionar en el filtro
- ID del botón: nombre del botón a validar si está visible.
- Índice de la Columna: número de la columna a validar.
- Nombre de la columna: valor a validar que exista con el mismo texto en el índice correcto de la tabla.
- ¿Columna editable? con la letra Y se valida si la columna se puede editar al validar un icono de lápiz al lado del nombre de la columna.

Como se muestra en la siguiente figura todos estos datos involucran múltiples funcionalidades como lo es la validación de los filtros, la selección de los datos en los filtros para la generación del reporte o de la pantalla (líneas de la 1 a la 4), la validación de los botones visibles (líneas de la 5 a la 7), la validación de las columnas de los datos generados (líneas de la 8 a la 14). Estos números de líneas varían en cada pantalla y en cada reporte. Certify ofrece la opción de utilizar el símbolo ^ para indicar que la variable o columna no se utiliza dependiendo de la funcionalidad en ejecución.

Recordset Editor

Name: *

Actuals - Adjust Last Available Cost

Description:

Records:

	Data_Input_Flag	Filter_Name	Data_Check	Filter_Selection	Filter_Label	Filter_radio_button	Filter_Value	button_id	Button_Mode	ColumnIndex	Column Name	Column_Editable	Dynamic_Flag
1	Y	Period	Y	Select One	^	^	2019/06	^	^	^	^	^	^
2	Y	Division_Overhead	N	^	^	^	^	^	^	^	^	^	^
3	Y	Profit_Center	Y	Select Multiple	^	^	2129	^	^	^	^	^	^
4	Y	Cost_Group	Y	Select Multiple	^	^	CAVE CREEK|NV|0|KB	^	^	^	^	^	^
5	N	^	N	^	^	^	^	btnExport	^	^	^	^	^
6	N	^	N	^	^	^	^	btnImport	^	^	^	^	^
7	N	^	N	^	^	^	^	btnClose	^	^	^	^	^
8	N	^	N	^	^	^	^	^	^	3	PC	N	N
9	N	^	N	^	^	^	^	^	^	4	Cost Group	N	N
10	N	^	N	^	^	^	^	^	^	5	Mfg Loc	N	N
11	N	^	N	^	^	^	^	^	^	6	Mfg Stage	N	N
12	N	^	N	^	^	^	^	^	^	7	LAC $	N	N
13	N	^	N	^	^	^	^	^	^	8	Adjusted value $	N	N
14	N	^	N	^	^	^	^	^	^	9	Reason for	N	N

ILUSTRACIÓN 4 - DATOS DE LOS ELEMENTOS DE LA PANTALLA EN ESPECÍFICO. ELABORACIÓN PROPIA.

Asimismo, es necesario mencionar que existe un recordset más el cual nos ayuda a navegar a la pantalla o reporte a probar. Este recordset contiene los siguientes datos:

- Aplicación: ICOST posee dos secciones Actuals (actual) y Forecast (pronóstico). Es necesario indicar cuál módulo se quiere utilizar.
- Nombre de la pantalla o reporte.
- Menú 0: este es el primer nivel del menú. En este caso Screens (pantallas) o Reports (reportes)

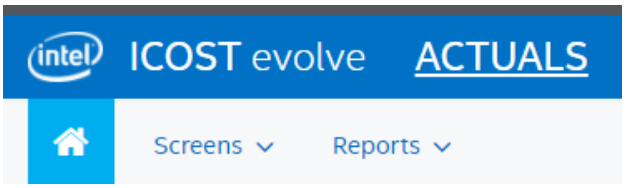


ILUSTRACIÓN 5 - VISTA DEL NIVEL 0 DEL MENÚ. ELABORACIÓN PROPIA.

- Menú 1: este es el segundo nivel del menú. En el caso de los reportes existen dos opciones ya sea Power BI o SSRS para efectos del proyecto nos enfocamos en los reportes SSRS debido que son páginas web y no involucra aplicaciones externas de reportería. Para las

pantallas este nivel de menú se refiere al área de ICOST como se puede ver en la imagen de abajo a la derecha.

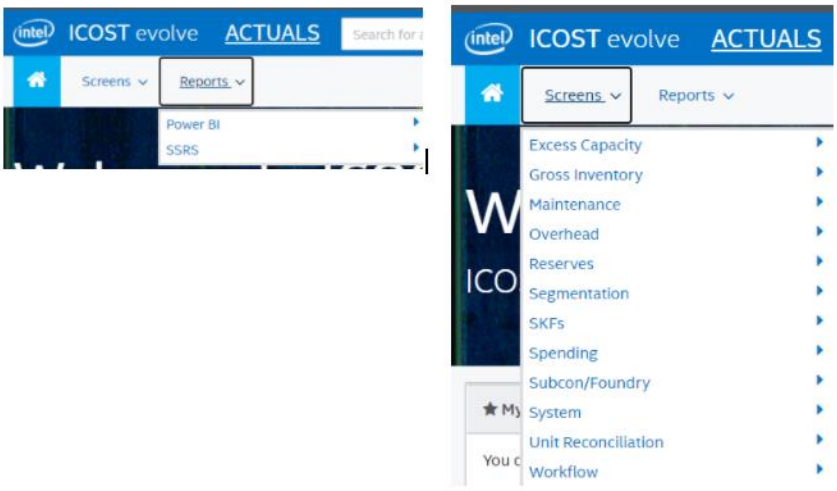


ILUSTRACIÓN 6 - MENÚ DE REPORTES Y PANTALLAS DE PRIMER NIVEL. ELABORACIÓN PROPIA.

- Menú 2: este nivel de menú nos permite en algunos casos como en las pantallas elegir la pantalla a probar. En el caso de los reportes se refiere al área de ICOST donde se encuentra el mismo.

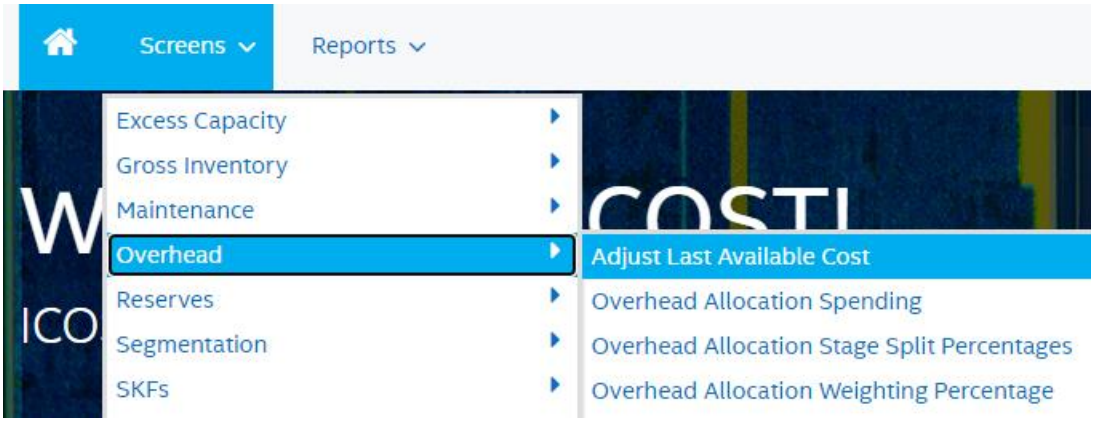


ILUSTRACIÓN 7 - MENÚ DE LAS PANTALLAS VISTA COMPLETA. ELABORACIÓN PROPIA.

- Menú 3: este nivel de menú en el caso de los reportes nos permite elegir el reporte a probar. En las pantallas aplica para casos especiales en los que en el menú debemos navegar a categorías más extensas.

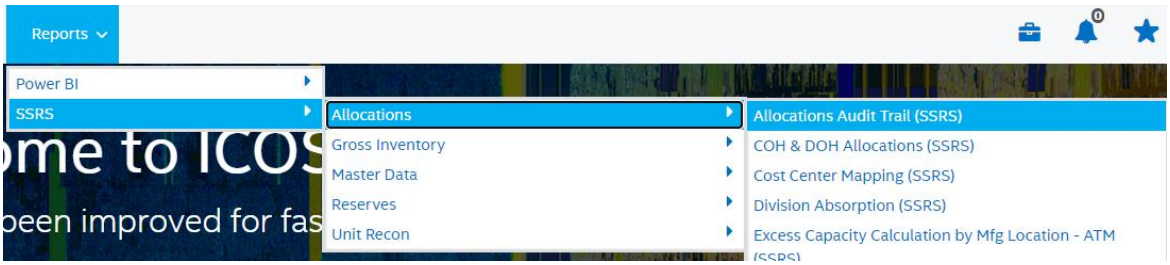


ILUSTRACIÓN 8 - MENÚ REPORTES SSRS VISTA COMPLETA. ELABORACIÓN PROPIA.

- Menú 4: este es un nivel extra para cubrir casos especiales en los que hay que navegar a categorías más extensas.
- ID de la pantalla o reporte: este ID lo podemos encontrar en el URL de la pantalla o reporte.

ILUSTRACIÓN 9 - ID DE LA PANTALLA. ELABORACIÓN PROPIA.

- Texto de búsqueda: este texto se utiliza para usar la funcionalidad del buscador de la página web con el fin de navegar a la pantalla o reporte.

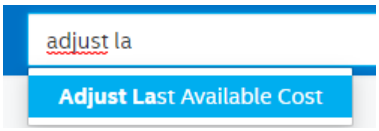


ILUSTRACIÓN 10 - TEXTO PARA SER UTILIZADO EN EL BUSCADOR. ELABORACIÓN PROPIA.

El recordset se muestra a continuación:

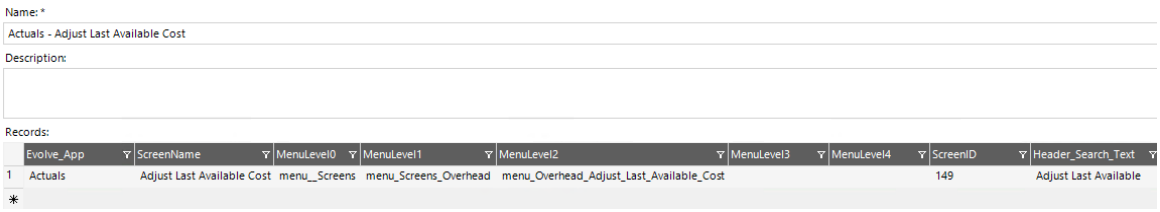


ILUSTRACIÓN 11 - DATOS DE LA PANTALLA EN ESPECÍFICO. ELABORACIÓN PROPIA.

Estas son las funcionalidades desarrolladas para la cobertura de diversas pruebas tanto en pantallas como en reportes SSRS:

- Validación de seguridad
    - Acceso Válido: este proceso utiliza varios subprocesos reutilizables que hemos desarrollado para un desarrollo de pruebas automatizadas eficientes, productivas y rápidas. Dichos subprocesos los vamos a explicar más adelante, como se puede ver en la imagen utilizamos para el proceso de validación de acceso válido:
        1. Abrir Google Chrome y cargar la página web de ICOST
        2. Elegir el rol deseado en la página
        3. Buscar la pantalla o reporte desde el buscador
        4. Validar el URL contiene el ID y en la página contiene el título correcto de la pantalla o reporte
        5. Navegación del menú para abrir la pantalla o reporte
        6. Validar el URL contiene el ID y en la página contiene el título correcto de la pantalla o reporte
        7. Cerrar Google Chrome
        8. Publicar resultado a Rally

Narrative
"GIVEN a USER with access to EVOLVE Actuals has loaded the Actuals Home Page
Initialize T[AppPath] to T[Evolve_App] ""
Exec Process App_Launch at "First Step" "None" "" "None" ""
Exec Process App_Launch_Chrome at "First Step" "None" "" "None" ""
"AND has chosen a role that has access to the page"
Exec Process EVOLVE_Choose_Role at "First Step" Evolve_Roles T[Role_RSKey] "Read Only" ""
"WHEN the user searches for the screen
Initialize T[tempResultNotes] to "WHEN the user searches for the screen
Exec Process EVOVLE_Search_for_a_screen at "First Step" Evolve_Screens T[ScreenName_RSKey] "Read Only" ""
Verify T[TestVerdict] Is Equal To "passed"
Initialize T[tempResultNotes] to "Check for correct screen after search navigation" "" "" "" "" "" "" "" "" "" "" "" "" "
Exec Process EVOLVE_Basic_Screen_Check at "First Step" Evolve_Screens T[ScreenName_RSKey] "Read Only" ""
Verify T[TestVerdict] Is Equal To "passed"
"THEN the user is able to browses to the page in the menu structure and click on it to bring the page up."
Initialize T[tempResultNotes] to "THEN the user is able to browses to the page in the menu structure " "" "" "" "
Exec Process EVOLVE_Menu_Navigation at "First Step" Evolve_Screens T[ScreenName_RSKey] "Read Only" ""
Verify T[TestVerdict] Is Equal To "passed"
Wait "3" seconds
"AND clicking on the page in the menu Structure brings up the correct page"
Initialize T[tempResultNotes] to "AND clicking on the page in the menu Structure brings up the correct page" "
Exec Process EVOLVE_Basic_Screen_Check at "First Step" Evolve_Screens T[ScreenName_RSKey] "Read Only" ""
"EndTheTest"
Exec Process Browser_Close at "First Step" "None" "" "None" ""
Exec Process U_Post_Result_to_Rally at "First Step" "None" "" "None" ""
"END of ICOST Landing Page Verification"

ILUSTRACIÓN 12 - EJEMPLO DE SCRIPT VALIDACIÓN DE ACCESO. ELABORACIÓN PROPIA.

- Acceso denegado: este proceso valida que el usuario no tiene acceso a la pantalla o reporte:

    9. Abrir Google Chrome y cargar la página web de ICOST
    10. Elegir el rol deseado en la página
    11. Buscar la pantalla o reporte desde el buscador
    12. Validar que el mensaje de error se muestra correctamente: “Your current role does not have access to ...” El rol actual no tiene acceso a la pantalla o reporte.

13. Navegación del menú para abrir la pantalla o reporte. Validación que en el menú no existe la opción del reporte o pantalla en cuestión.

14. Cerrar Google Chrome

15. Postear resultado a Rally

    - Acceso de lectura: este proceso es muy similar al de validación de acceso válido puesto que abre la página web, se busca o se navega en el menú la pantalla o reporte, se valida el ID y título, cerrar Chrome y postear el resultado en Rally.

- Validación de los elementos del UI

    - Filtros: este subproceso valida cada filtro que existe dentro del reporte o pantalla bajo prueba, así mismo valida el tipo de filtro si es selección única, múltiple o radiobutton. Selecciona el valor en cada filtro y genera los datos de la pantalla o reporte.

    - Botones: valida que la pantalla o reporte contiene los botones especificados en el recordset.

    - Columnas: valida que la pantalla o reporte contiene las columnas correctas, en el orden correcto y si son editables o no.

- Validación de la existencia de datos: este proceso valida la existencia de datos con los filtros seleccionados al generar el reporte o pantalla específicamente se valida que no existe el mensaje de “no hay datos”.

- Validación del reporte histórico (solo para Reportes SSRS): este proceso guarda o guardó datos de meses anteriores en una carpeta compartida, con el fin de futuras ejecuciones con filtros seleccionando algún mes anterior, los datos deben ser iguales ya que al ser datos históricos no pueden cambiar ya que en finanzas los libros contables se cierran mes a mes y deben ser modificados.

Subprocesos reutilizables:

Estos subprocesos los consideramos necesarios para la realización del marco de trabajo ya que nos facilitó el desarrollo de todos los scripts en Certify.

- Abrir ICOST y seleccionar la aplicación (Actuals o Forecast): este sub proceso nos ayuda a abrir una página web en Google Chrome, asegurándose primeramente que no existan páginas abiertas, en el caso de que haya alguna página abierta el mismo script cierra todo y abre solamente la página bajo prueba en este caso la página de ICOST, el script construye el URL usando el recordset de los ambientes para tener la flexibilidad de abrir cualquier ambiente requerido además en el mismo URL se indica cuál módulo se desea abrir (Actuals o Forecast), maximiza la ventana, y espera unos segundos para que la página termine de cargar en su totalidad.

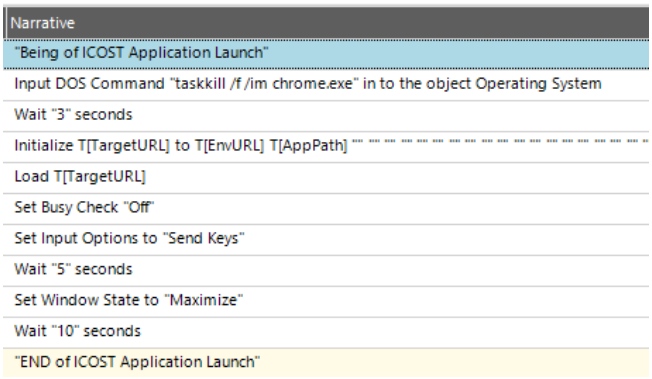


ILUSTRACIÓN 13 - SUBPROCESO PARA ABRIR CHROME "APP_LAUNCH_CHROME". ELABORACIÓN PROPIA.

- Esperar carga de la página: este es un pequeño subproceso con el fin de detectar el icono de "cargando" para que el script pueda esperar hasta que las ventanas de la página hayan cargado para continuar con los siguientes procesos.

- Elegir el rol: este subproceso nos ayuda a siempre elegir el rol deseado a la hora de abrir la página de ICOST, esta es una funcionalidad esencial puesto que es requerido elegir el rol con el que se va a probar las pantallas o reportes.

- Navegar a la pantalla o reporte: este subproceso lleva mayor complejidad debido que cada pantalla o reporte cuenta con distintos niveles del menú como lo vimos en la explicación

del recordset, existe del menú nivel 0 al 4. Este subproceso nos ayuda a navegar por el menú hasta el reporte o pantalla y abrirlo, de paso se verifica que el menú se comporte correctamente y que el reporte o pantalla exista en el menú.

- Buscar la pantalla o reporte: este subproceso fue desarrollado con el fin de probar el buscador y además para mayor facilidad de acceso hacia las pantallas o reportes.

- Cerrar Chrome: este es un pequeño subproceso que nos ayuda a cerrar Google Chrome a la hora de terminar de ejecutar el proceso.

- Publicar en Rally: este subproceso fue desarrollado con el fin de publicar en Rally los resultados de las pruebas automatizadas, involucra la tecnología llamada CURL y la línea de comandos para ejecutar el API de Rally y enviar la información correspondiente, es de suma importancia debido que permite la trazabilidad entre las pruebas automatizadas con los casos de pruebas documentados en Rally.

Otro gran aspecto de este marco de trabajo es el manejo de palabras clave o Keyword-Driven, anteriormente vimos cómo utilizando los recordset nos permite hacer pruebas Data-Driven o basadas en datos, ahora detallaremos más en detalle cómo se desarrolló a su vez este concepto. Tenemos recordsets en los cuales una columna es dedicada a una palabra clave definida para indicarle al script que debe ejecutar una acción (subproceso) en específico, esto permite ser más flexible y cubrir más escenarios, además de darle al usuario una manera amigable de automatizar sin tener que desarrollar un script, básicamente agregando una línea nueva en el recordset va a tener una prueba automatizada en la cuál va a poder elegir la funcionalidad a probar, la pantalla o reporte a probar entre otras acciones.

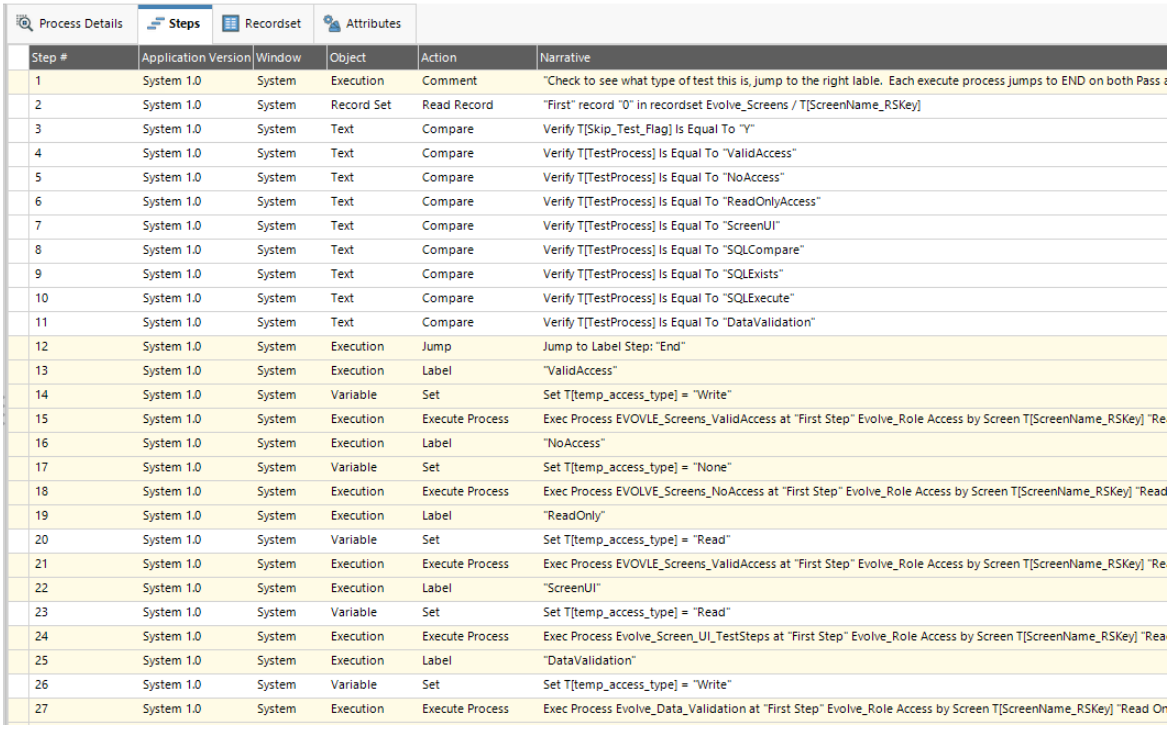
Al desarrollar este proceso, fue necesario validar dicha columna del recordset con el propósito de validar si es un texto de los disponibles, como se puede ver en la siguiente imagen, este proceso de las pantallas tenemos varias palabras claves disponibles, se muestran en inglés puesto que en Intel todo debe realizarse en dicho idioma:

- ValidAccess: si la palabra clave es esta va a ejecutar el proceso EVOLVE_*Screens_ValidAccess* para validar que el usuario posee acceso.

- NoAccess: se ejecuta el proceso *EVOLVE_Screens_NoAccess* para validar que el usuario no posee acceso.
- ReadOnlyAccess: se ejecuta el proceso *EVOLVE_Screens_ValidAccess* con la diferencia que con una variable se espeicifica validar específicamente el acceso de lectura.
- ScreenUI: se ejecuta el proceso *Evolve_Screen_UI_TestSteps* para validar los elementos de la interfaz gráfica de la pantalla.

En el caso de los reportes son las siguientes:

- SSRSReport UI: se ejecuta el proceso *SSRS_Report_UI_Validations* para validar los elementos de la interfaz gráfica del reporte.
- HistoricalCSVCompare: se ejecuta el proceso *SSRS_Reports_Export_as_CSV_Validate_CSV_file* para la validación de los datos históricos.
- ValidAccess: se ejecuta el proceso *EVOLVE_SSRS_Report_ValidAccess* para validar que el usuario posee acceso.
- NoAccess: se ejecuta el proceso *EVOLVE_SSRS_Report_NoAccess* para validar que el usuario no posee acceso.



Process Details | Steps | Recordset | Attributes

Step #	Application Version	Window	Object	Action	Narrative
1	System 1.0	System	Execution	Comment	"Check to see what type of test this is, jump to the right lable. Each execute process jumps to END on both Pass
2	System 1.0	System	Record Set	Read Record	"First" record "0" in recordset Evolve_Screens / T[ScreenName_RSKey]
3	System 1.0	System	Text	Compare	Verify T[Skip_Test_Flag] Is Equal To "Y"
4	System 1.0	System	Text	Compare	Verify T[TestProcess] Is Equal To "ValidAccess"
5	System 1.0	System	Text	Compare	Verify T[TestProcess] Is Equal To "NoAccess"
6	System 1.0	System	Text	Compare	Verify T[TestProcess] Is Equal To "ReadOnlyAccess"
7	System 1.0	System	Text	Compare	Verify T[TestProcess] Is Equal To "ScreenUI"
8	System 1.0	System	Text	Compare	Verify T[TestProcess] Is Equal To "SQLCompare"
9	System 1.0	System	Text	Compare	Verify T[TestProcess] Is Equal To "SQLExists"
10	System 1.0	System	Text	Compare	Verify T[TestProcess] Is Equal To "SQLExecute"
11	System 1.0	System	Text	Compare	Verify T[TestProcess] Is Equal To "DataValidation"
12	System 1.0	System	Execution	Jump	Jump to Label Step: "End"
13	System 1.0	System	Execution	Label	"ValidAccess"
14	System 1.0	System	Variable	Set	Set T[temp_access_type] = "Write"
15	System 1.0	System	Execution	Execute Process	Exec Process EVOVLE_Screens_ValidAccess at "First Step" Evolve_Role Access by Screen T[ScreenName_RSKey] "Re
16	System 1.0	System	Execution	Label	"NoAccess"
17	System 1.0	System	Variable	Set	Set T[temp_access_type] = "None"
18	System 1.0	System	Execution	Execute Process	Exec Process EVOLVE_Screens_NoAccess at "First Step" Evolve_Role Access by Screen T[ScreenName_RSKey] "Read
19	System 1.0	System	Execution	Label	"ReadOnly"
20	System 1.0	System	Variable	Set	Set T[temp_access_type] = "Read"
21	System 1.0	System	Execution	Execute Process	Exec Process EVOVLE_Screens_ValidAccess at "First Step" Evolve_Role Access by Screen T[ScreenName_RSKey] "Re
22	System 1.0	System	Execution	Label	"ScreenUI"
23	System 1.0	System	Variable	Set	Set T[temp_access_type] = "Read"
24	System 1.0	System	Execution	Execute Process	Exec Process Evolve_Screen_UI_TestSteps at "First Step" Evolve_Role Access by Screen T[ScreenName_RSKey] "Rea
25	System 1.0	System	Execution	Label	"DataValidation"
26	System 1.0	System	Variable	Set	Set T[temp_access_type] = "Write"
27	System 1.0	System	Execution	Execute Process	Exec Process Evolve_Data_Validation at "First Step" Evolve_Role Access by Screen T[ScreenName_RSKey] "Read On

ILUSTRACIÓN 14 - PROCESO PARA PANTALLAS. ELABORACIÓN PROPIA.

Todos estos procesos están contenidos en los llamados Controladores o “Controllers” en Certify, dichos nos permite organizar los subprocesos en un solo proceso a conveniencia del usuario. Hemos creado varios controladores para separar las funcionalidades con el fin de que sean legibles y fáciles de mantener durante el tiempo.

Controladores

ICOST_Evolve_Weekly_Regression_Screen_Data_Validation
ICOST_Evolve_Weekly_Regression_Screen_Security
ICOST_Evolve_Weekly_Regression_Screen_UI_Checkout
ICOST_Evolve_Weekly_Regression_SSRS_Reports

Procesos dentro del Controlador de Seguridad de las pantallas

"Start of LO Process"
Exec Process EVOLVE_ScreenTestDriver at "First Step" Evovle_Screen_Tests PS_NoAccess_RoleTest_Regression_Suit "Rea
Exec Process EVOLVE_ScreenTestDriver at "First Step" Evovle_Screen_Tests PS_ReadAccess_RoleTest_Regression_Suit "R
Exec Process EVOLVE_ScreenTestDriver at "First Step" Evovle_Screen_Tests PS_ValidAccess_RoleTest_Regression_Suit "R
"END of LO process"

Recorset de un proceso (Accceso no válido para pantallas)

Records:

	Skip_Test_Flag	ScreenName_RSKey	TestCaseID	TestCaseObjectID	U_TestSetObjectID	TestProcess	Test_Type_Filter	DataGridEmpty	TestVerdic
1		Actuals - Activity	TC32206	235944189436	257351946872	NoAccess	Smoke		Pass
2		Actuals - Adjust Last	TC32491	235943704628	257351946872	NoAccess	Smoke		Pass
3		Actuals - Allocate	TC32552	235944181336	257351946872	NoAccess	Smoke		Pass
4		Actuals - Cost Center	TC32542	235944178256	257351946872	NoAccess	Smoke		Pass
5		Actuals - Cost per Unit	TC32528	235944174484	257351946872	NoAccess	Smoke		Pass
6		Actuals - Costing	TC32390	235943676696	257351946872	NoAccess	Smoke		Pass
7		Actuals - Demand and	TC32513	235943710864	257351946872	NoAccess	Smoke		Pass
8		Actuals - Direct	TC32208	235943140648	257351946872	NoAccess	Smoke		Pass
9		Actuals - Div Eng	TC32555	235944182340	257351946872	NoAccess	Smoke		Pass
10		Actuals - Div Eng	TC32549	235944180560	257351946872	NoAccess	Smoke		Pass
11		Actuals - Div Eng TD	TC32558	235944183072	257351946872	NoAccess	Smoke		Pass
12		Actuals - DPW	TC32522	235943713588	257351946872	NoAccess	Smoke		Pass
13		Actuals - Excess	TC32154	235943119532	257351946872	NoAccess	Smoke		Pass
14		Actuals - Excess	TC32561	235944183844	257351946872	NoAccess	Smoke		Pass
15		Actuals - Fab Ex Cap	TC32157	235943120384	257351946872	NoAccess	Smoke		Pass
16		Actuals - Forecasted	TC32225	235943147356	257351946872	NoAccess	Smoke		Pass

ILUSTRACIÓN 15 - EJEMPLOS DE CONTROLADORES EN CERTIFY. ELABORACIÓN PROPIA.

Nivel unitario - Parasoft SOATest

SOATest es la herramienta que se seleccionó para realizar las pruebas unitarias del proyecto ICOST, después de varias reuniones con personal involucrado en Icost y con expertos de la herramienta, se decidió que de las herramientas disponibles actualmente en Intel, SOATest sería la que mejor encajaría con el proyecto ICOST y sus pruebas unitarias.

Esta herramienta es utilizada para las creación de pruebas y análisis de pruebas de API, tiene diferentes funcionalidades para realizar las pruebas, estas variadas funcionalidades fueron las que se vieron muy útil para la situación del proyecto, como que permite almacenar las respuestas de una consulta para ser utilizadas posteriormente, realizar llamados de otras pruebas en la aplicación para ser utilizadas dentro de otra prueba, nos permite creación de variables, conexión con diferentes bases de datos, creación de diferentes ambientes de pruebas, permite realizar actualizaciones del estado de un caso de prueba en la herramienta Rally, la cual es usada en Intel para el manejo de proyectos, y además una parte muy importante que se tomó en cuenta es que permite realizar consultas SQL y no solamente de API.

Para la implementación de las pruebas unitarias era necesario realizar las consultas de SQL, por lo que SOATest fue la mejor opción, la flexibilidad que se tuvo para realizar estas pruebas unitarias fue bastante alta, pues en ningún punto de la historia del proyecto ICOST se había realizado este tipo de pruebas, por lo que se pudieron adaptar a lo que se necesitara de acuerdo a los datos recolectados de las entrevistas y reuniones con los desarrolladores y analistas del sistema.

El proceso automatizado de pruebas unitarias se dividió en tres partes diferentes cada prueba, para lograr una prueba exitosa se ocupa cumplir con tres requerimientos, el primero es que dependiendo de la operación a ejecutar en SQL anteriormente son necesarios datos preexistente en la base de datos, como segundo requerimiento es la ejecución de la consulta SQL y como tercer requerimiento es la eliminación de los datos predefinidos para la prueba, cada uno de estos pasa tienen un propósito específico.

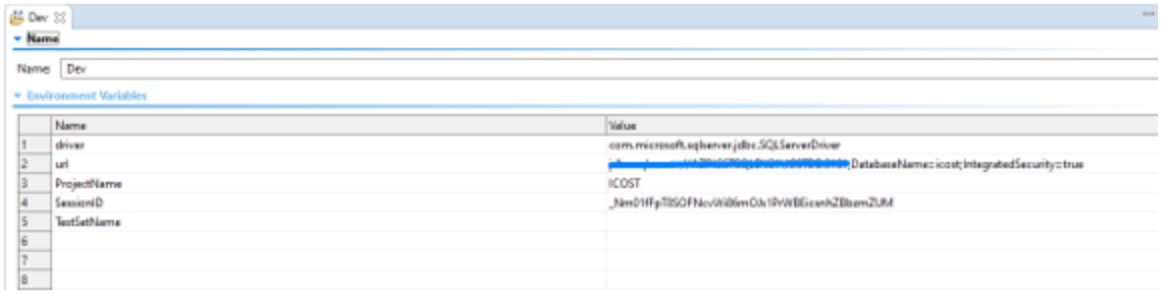


ILUSTRACIÓN 16 - VARIABLES DEFINIDAS PARA EL AMBIENTE DE DESARROLLO.

SoaTest cuenta con la funcionalidad mencionada anteriormente de definir diferentes ambientes de pruebas, en estos ambientes se generan variables que serán utilizadas, con el fin de facilitar el proceso de actualización de variables en caso de que fuera necesario, como por ejemplo que haya un cambio en la conexión de la base de datos y de esta forma se actualiza globalmente sin tener que cambiarlo prueba por prueba.

La primera parte de la prueba que se automatizó en SOATest es la de insertar información preexistente necesaria para la ejecución de la prueba que se desea realizar, en algunos casos no es necesario que existan los datos anteriormente a la prueba por lo que no siempre se va a ejecutar esta parte de la prueba, el principal problema de que se ejecutara la prueba sin los datos es que tiene una alta probabilidad de fallar, ya que se utiliza una base de datos relacional y los requerimientos de llaves primarias y llaves foráneas deben cumplirse.

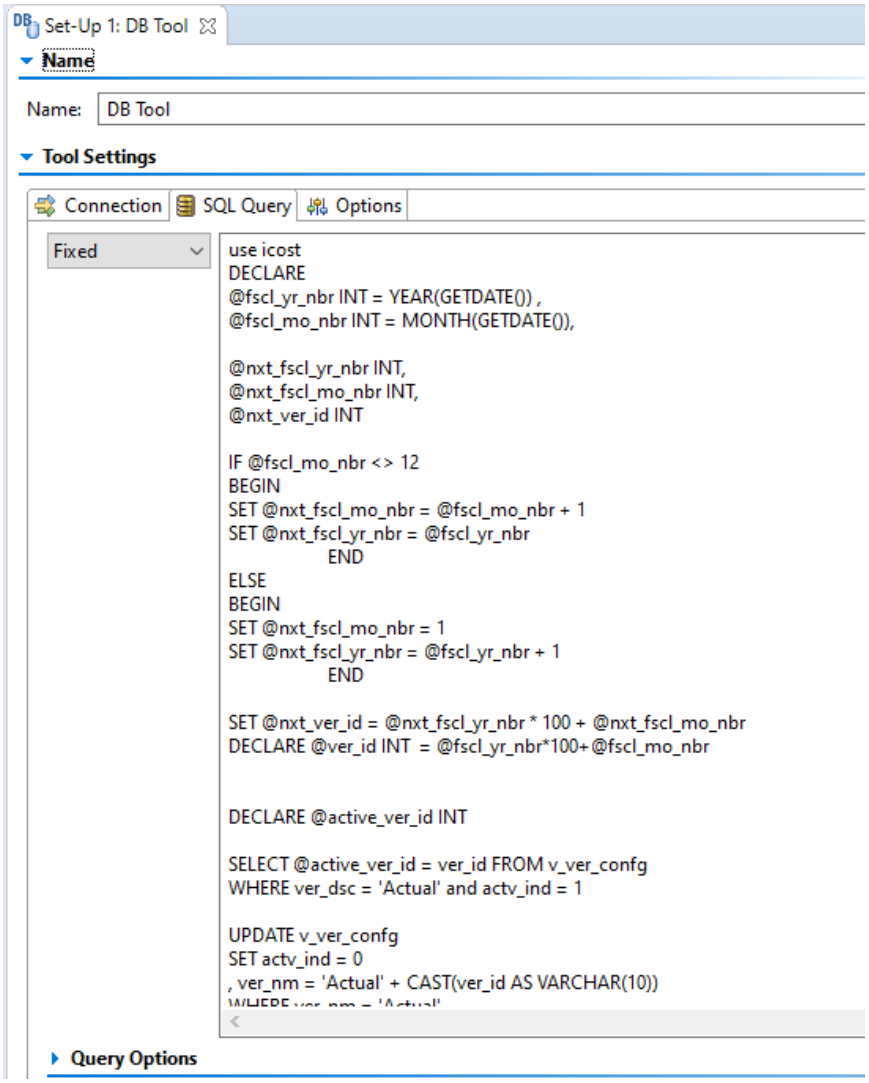


ILUSTRACIÓN 17 - SET-UP TEST, LLAMADO A LA CONSULTA SQL DE PREPARACIÓN

Como se muestra en la imagen la primer parte de la prueba se llama Set-Up Test que como su nombre lo indica es una prueba de preparación, de igual manera que la parte principal de la prueba es una consulta directa a la base de datos en la cual se insertan los diferentes datos necesarios, el nombre del Set-Up Test en este caso específico se le dio un nombre genérico que represente su funcionalidad, Insert Dummy Records .

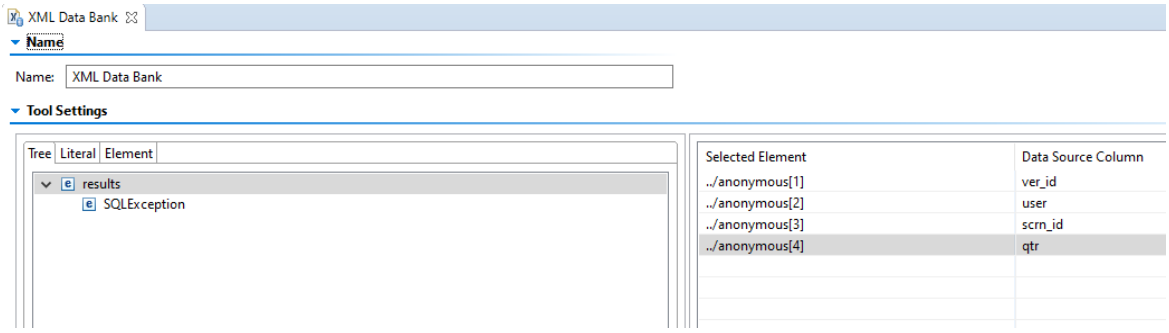


ILUSTRACIÓN 18 - FUNCIONALIDAD XML DATA BANK, ALMACENAMIENTO DE RESULTADOS COMO VARIABLES

La imagen anterior muestra que del Set-Up Test puede generar resultados necesarios en la prueba principal, por lo que gracias a la funcionalidad de la SOATest podemos almacenar esta información en variables que pueden ser utilizadas en el mismo escenario de prueba, el resultado es entregado en formato XML por lo que se utiliza la funcionalidad XML Data Bank que es la que permite almacenar en variables el resultado.

La siguiente parte es la prueba principal, en esta se ejecuta la consulta SQL, en este caso más específicamente el procedimiento almacenado que se necesitaba probar, en esta prueba se hace uso de variables definidas en el proceso anterior o como variables globales, la consulta ejecutada es básicamente el caso de prueba que se identificó durante las reuniones y entrevistas a lo largo de la ejecución del proyecto.

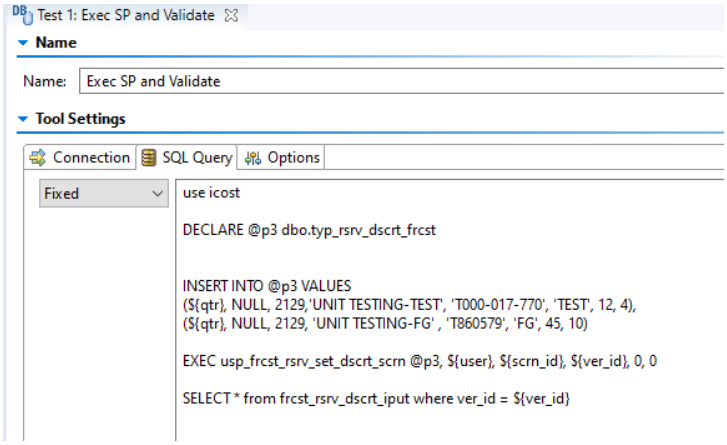


ILUSTRACIÓN 19 - DEFINICIÓN DE CONSULTA SQL A EJECUTAR.

Las variables se llaman utilizando el siguiente formato ${variable} así como se hizo el llamado en la prueba a ejecutar.

Como método para probar que el proceso se haya ejecutado correctamente y que el resultado sea el esperado se utilizó la funcionalidad de XML Assertor proporcionada por la herramienta SOA Test, con esta funcionalidad podemos definir las condiciones con las cuales podemos asegurar si el proceso funcionó o hubo algún error, por ejemplo si el resultado fue una columna vacía y se esperaba un resultado específico o que estuviera con al menos un dato la prueba va a fallar y decir el porqué del fallo.

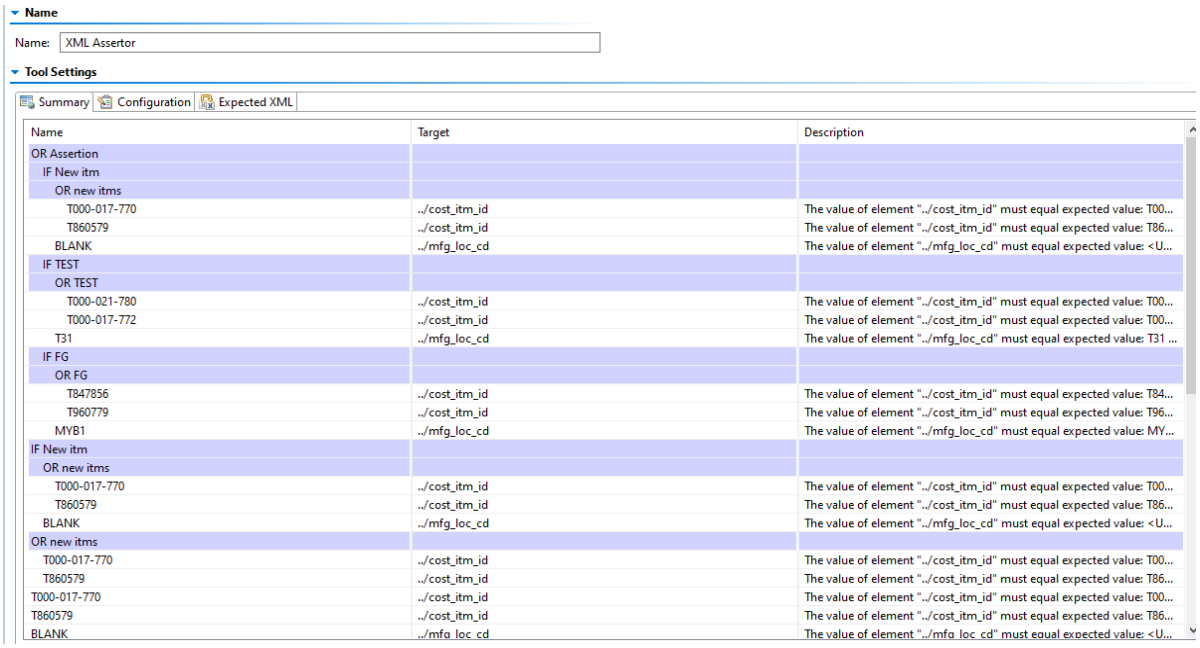


ILUSTRACIÓN 20 - XML ASSERTOR CON LAS CONDICIONES PARA APROBAR UNA PRUEBA.

Después de la ejecución de la prueba principal se continuó con la prueba Tear Down, que es el proceso en el que se eliminaron todos los datos que se insertaron para realizar la prueba, esto con el fin de no dejar información extra o “basura” en la base de datos.

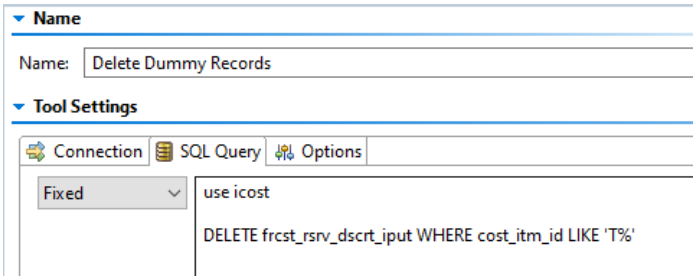


ILUSTRACIÓN 21 - ELIMINACIÓN DE DATOS DE PRUEBA.

Además, se realizó el llamado a la API de Rally para poder actualizar los resultados del caso de prueba en Rally correspondientemente al resultado obtenido por la prueba unitaria automatizada, para esto es necesario el API de Rally, la llave de la API (sirve como forma de autenticación), número del proyecto y número de caso de prueba. Las pruebas tienen solamente dos posibles resultados, éxito o fallo, esto nos ayuda a llevar un control del progreso de las pruebas a tiempo real y adicionalmente el principal beneficio es la trazabilidad de cada una de las automatizaciones que va a ir creciendo durante el tiempo y la documentación en Rally.

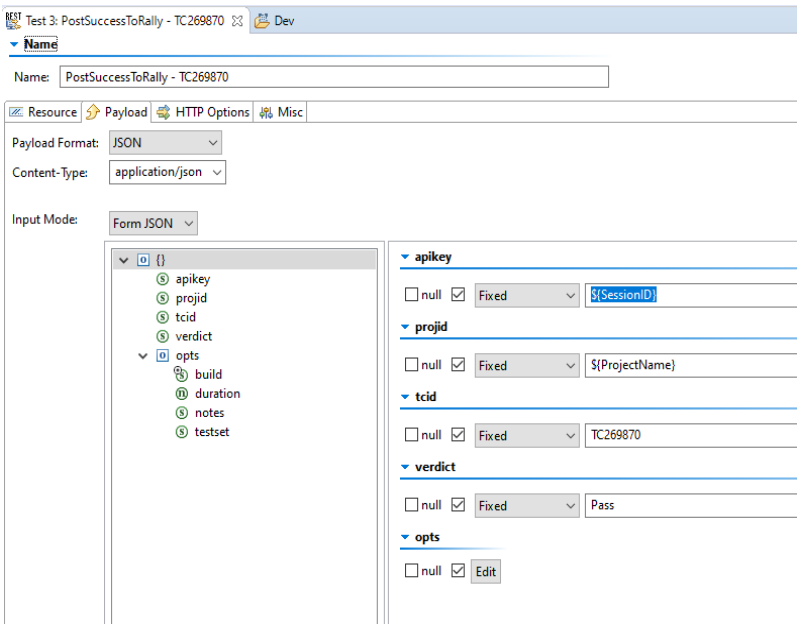


ILUSTRACIÓN 22 - CONFIGURACIÓN LLAMADA DE API DE RALLY PARA PUBLICAR LOS RESULTADOS.

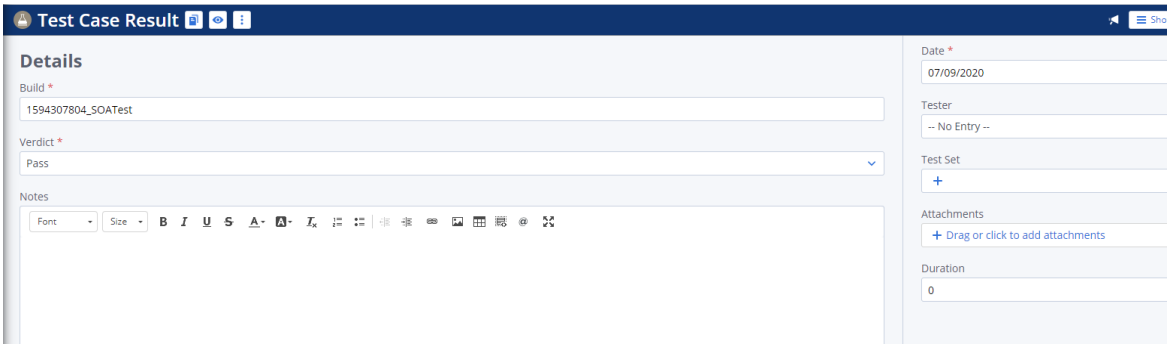


ILUSTRACIÓN 23 - RESULTADO PUBLICADO POR SOATEST EN RALLY.

Este sería el flujo completo que se implementó para la ejecución de pruebas unitarias utilizando SOA Test,como punto extra se implementó el proceso de CICD por medio de GitLab, esto con el fin de ejecutar pruebas de regresión cada vez que ocurre un cambio en las pruebas. Para la ejecución de este paso se obtuvo acceso a una máquina virtual en la cual se realizaron la instalación de características necesarias para ponerlo en marcha, las pruebas se ejecutan automáticamente en la máquina virtual.

- Runner de Gitlab
- SOA Test
- Creación de script, para ser ejecutado por el Runner
- Configuración de SOA Test específica para la integración de CICD

Se realizó la configuración necesaria del lado de GitLab, inicialización del Runner, configuración del archivo yml que se encuentra en el repositorio y accesos al proyecto de GitLab a los desarrolladores y analistas que estarían a cargo.

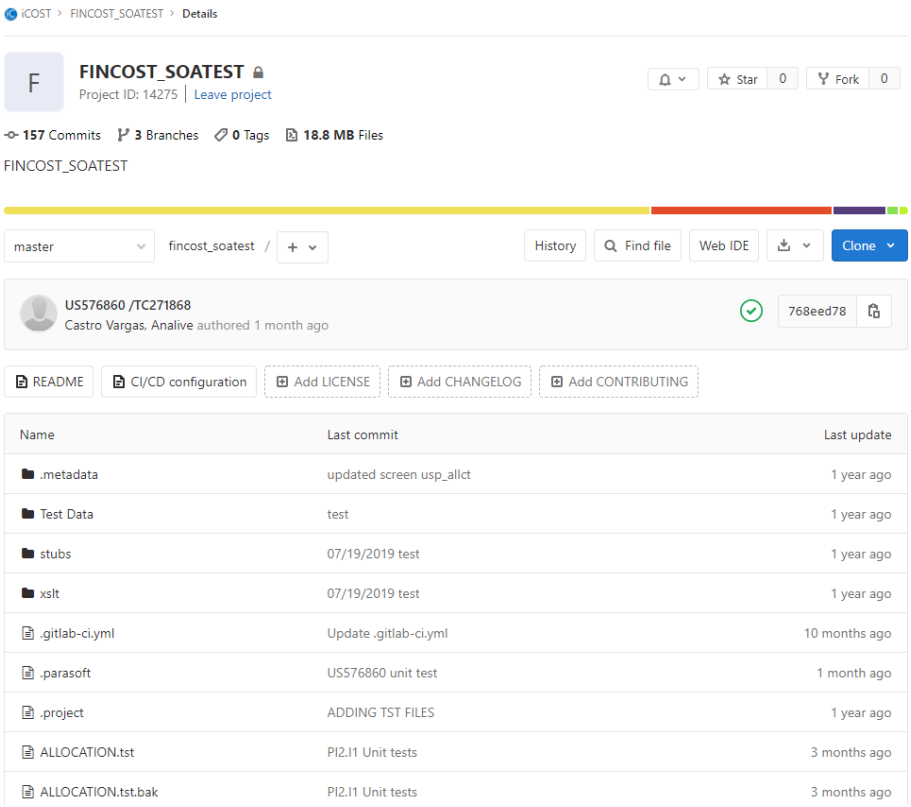


ILUSTRACIÓN 24 - REPOSITORIO GITLAB DONDE SE ALOJA EL PROYECTO AUTOMATIZADO DE PRUEBAS UNITARIAS EN SOATEST.

Las condiciones para hacer uso de la integración serían, que los que realicen las pruebas tengan Git o alguna herramienta afín al control de versiones y GitLab instalada, saber utilizar Git, acceso al repositorio de GitLab y el proyecto con los cambios actualizados; los encargados actuales cuentan con conocimiento de Git y GitLab además de que son las herramientas utilizadas actualmente en Intel por lo que se encuentra en cumplimiento.

Otro aspecto de gran importancia es la implementación de una base de datos en este caso en MongoDB la cual es una base de datos multiplataforma orientada a documentos. Intel cuenta con los recursos y el procedimiento establecido para crear nuevas bases de datos en diversas tecnologías dentro de las cuales se encuentra MongoDB.

Con esta integración de la base de datos en MongoDB y SOATest se busca obtener los beneficios del concepto Data-Driven al igual que en el nivel de interfaz con Certify. a continuación, se

muestra la base de datos creada, como se puede observar utilizamos un cliente llamado MongoDB Compass para mayor facilidad de acceso y cuenta con una interfaz amigable al usuario.

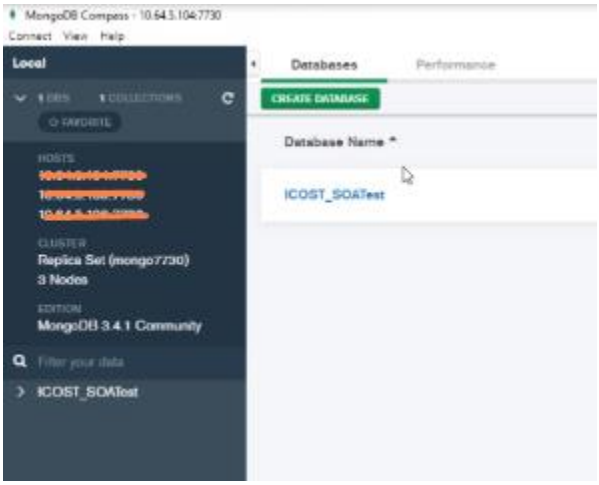


ILUSTRACIÓN 25 - HERRAMIENTA MONGODB COMPASS

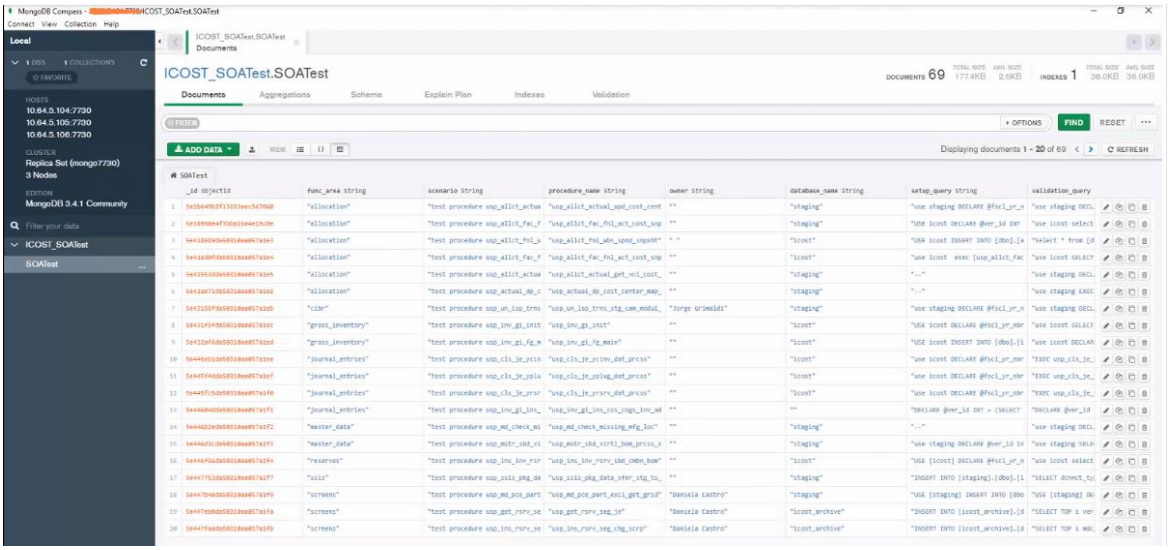


ILUSTRACIÓN 26 - BASE DE DATOS ICOST SOATEST

Dicha base de datos cuenta con las siguientes columnas:

- Área funcional: esto nos indica el área de Icost al cual pertenece esta prueba y ayuda a la organización de las pruebas, podemos filtrar por área y ejecutar pruebas más focalizadas.

- Escenario: facilita la comprensión de la prueba para un mejor entendimiento y mantenimiento de las pruebas a futuro.
- Nombre del procedimiento almacenado
- Dueño o encargado de la prueba
- Nombre de la base de datos: este dato es importante ya que nos ayuda a ubicar los procedimientos almacenados ya que ICOST posee múltiples bases de datos.
- Setup query: el SQL a ejecutar al inicio de la prueba.
- Validation query: el SQL para validar los datos
- Teardown query: el SQL a ejecutar al final de la prueba para limpiar los datos utilizados.
- Validación: es el resultado esperado de la prueba en formato XML
- Test case o caso de prueba: es el ID de rally
- Run: es un indicador para establecer si correr o no la prueba.

Estos son los datos necesarios para crear nuevas pruebas tan sólo llenando dichas columnas en la base de datos, sin conocimiento técnico en la herramienta SOATest, no es requerido que tan siquiera abra la herramienta para crear una prueba, ya que al nosotros crear un script genérico en SOATest conectado a esta base de datos, facilitamos dicha creación de pruebas automatizadas dirigidas por datos. Además, el mantenimiento de las pruebas debe realizarse solamente en la base de datos y evitamos la navegación por todos los scripts en la herramienta SOATest ya que a futuro podría volverse incontrolable debido a la cantidad de procedimientos almacenados existentes.

A continuación, se muestra el script genérico creado para sostener este concepto Data-Driven junto a MongoDB y siguiendo el marco de trabajo explicado anteriormente

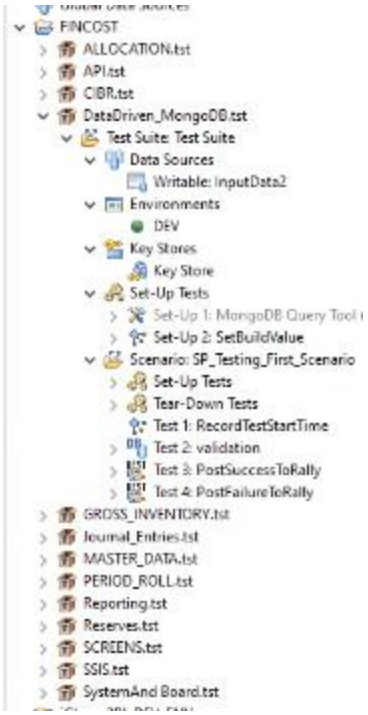


ILUSTRACIÓN 27 - ESTRUCTURA EN SOATEST INCLUYENDO LA INTEGRACIÓN CON MONGODB

Podemos ver una serie de configuraciones como lo es tener un DataSource, un ambiente configurado, setups, teardowns, validaciones, registro del tiempo de duración y el posteo de los resultados a Rally.

El DataSource creado extrae de la base de datos en MongoDB los datos necesarios para la ejecución de las pruebas, las carga una sola vez al inicio para así tener en memoria todas las pruebas a ejecutar, esto ayuda a tener un mejor rendimiento ya que no debemos hacer múltiples consultas a la base de datos.

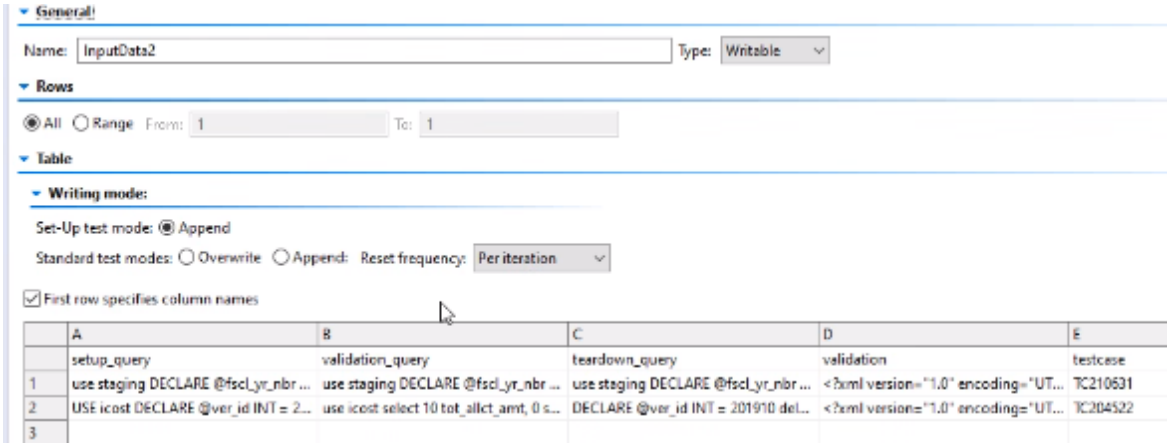


ILUSTRACIÓN 28 - SOATEST DATASOURCE INICIAL

También se configuró un ambiente de desarrollo el cual contiene varias variables explicadas anteriormente con la diferencia que en este caso se incluyen los datos de la base de datos de MongoDB. Los ambientes nos permiten crear varios y elegir cuales usar, por ejemplo, desarrollo, preproducción y producción.

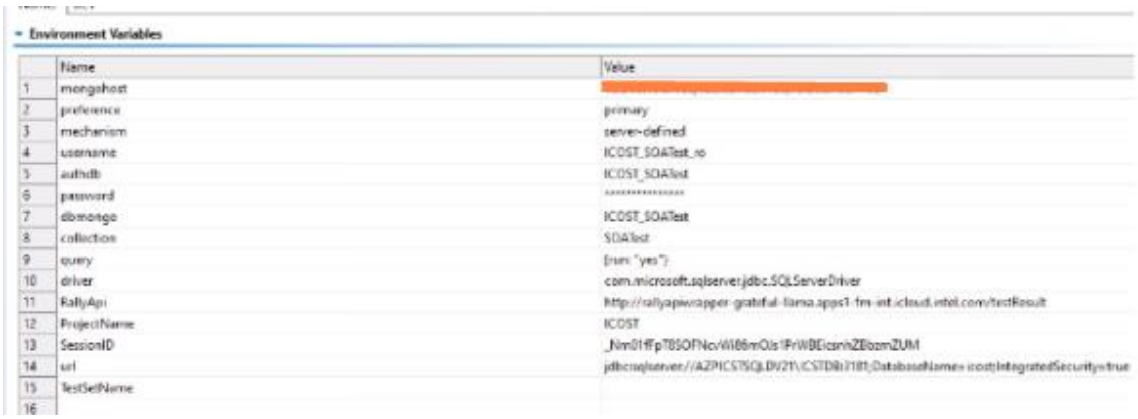


ILUSTRACIÓN 29 - SOATEST VARIABLES DE AMBIENTE

Los setup tests y teardown tests específicamente en la sección del query de SQL en lugar de crear múltiples tests para diversas pruebas, tenemos solamente una con la diferencia que esta es parametrizada con la variable del DataSource el cual contiene la información extraída de la base de datos en MongoDB.

ILUSTRACIÓN 30 - PRUEBAS PARAMETRIZADAS

Asimismo, la validación de las pruebas se hace por medio de lo que se conoce como Diff en SOAtest, facilita la validación de los resultados de una consulta en SQL contra un valor predeterminado en formato XML en este caso. El cual a su vez se encuentra parametrizado con la variable validación de la base de datos.

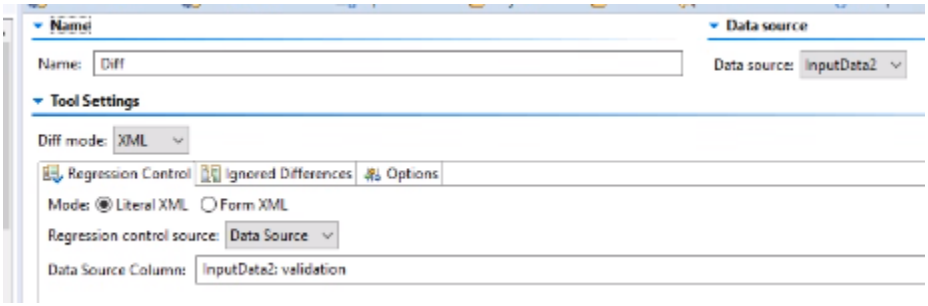


ILUSTRACIÓN 31 - VALIDACIÓN PARAMETRIZADA

Nivel de servicio - Parasoft SOATest

A nivel de servicio se realizaron prototipos para la implementación de un marco de trabajo como referencia para futuras automatizaciones de este nivel, esto debido que ICOST no cuenta con suficientes APIs al momento de este proyecto, para realizar pruebas concretas de las API se necesitan casos de prueba definidos, por lo tanto, se utilizaron dos o tres APIs disponibles al momento para construir este marco de trabajo. Se comprobó la automatización de dichas pruebas por medio de la herramienta SOATest, ya que dicha herramienta es especialmente diseñada para este nivel de pruebas.

La principal propuesta de realizar estas pruebas es demostrar que el proyecto tiene escalabilidad a nivel de servicio, y sentar bases para futuras implementaciones, con esto se brinda un marco de trabajo utilizando prototipos de pruebas automatizadas de cómo se deberían de automatizar las

pruebas de APIs una vez estén lista para estar en funcionamiento y ya con este proceso adelantado se tiene un patrón definido a seguir una vez finalizado el desarrollo de la API por lo que se facilitan las pruebas de estas.

El proceso para las pruebas de APIs que se realizó fue el siguiente:

- Se crea un archivo .tst en SOATest
- De igual manera a las pruebas de procedimientos almacenados se definen ambientes de trabajo con sus respectivas variables
- Se crea una Prueba de Cliente Rest
- Se define el método/operación que va a realizar la API (GET, POST, DELETE...)
- En URL se define el URL que se utiliza para la llamada de la API
- En caso de que la API lo requiera se define el Payload que es la información necesaria para que la API realice la operación respectiva de su método
- Se definen las cabeceras de la API si es necesario
- Se introducen las credenciales de autorización
- Se agregan funcionalidades de almacenamiento de variables o de comprobación de resultados

Esto brinda al proyecto un amplio margen de escalabilidad, pues permite nuevas formas de realizar pruebas y automatización dentro del marco de trabajo y con herramientas que ya son utilizadas y se tiene experiencia con estas.

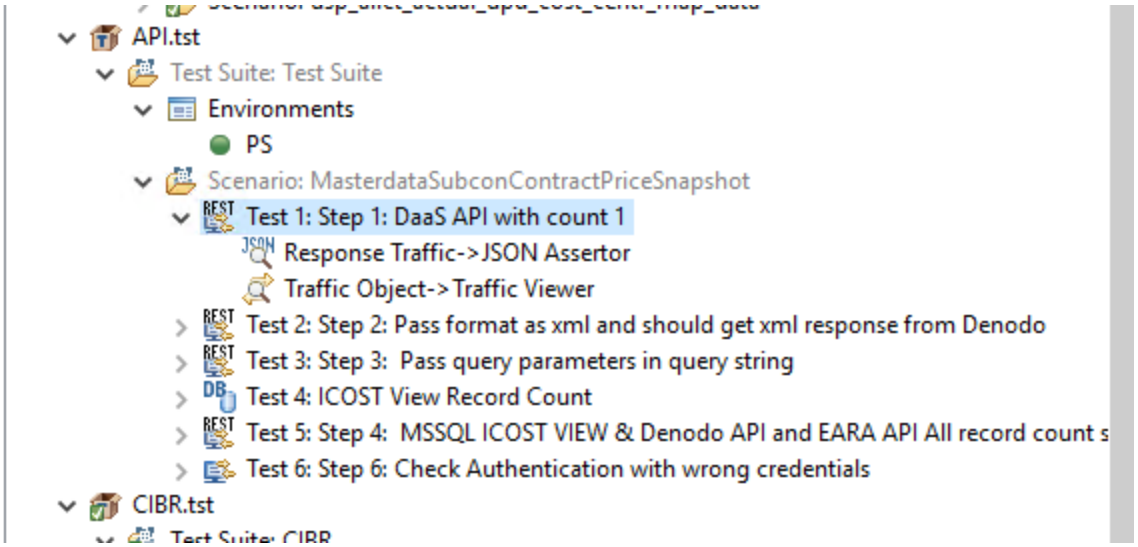


ILUSTRACIÓN 32 - PROTOTIPOS DE PRUEBAS DE APIS CREADAS

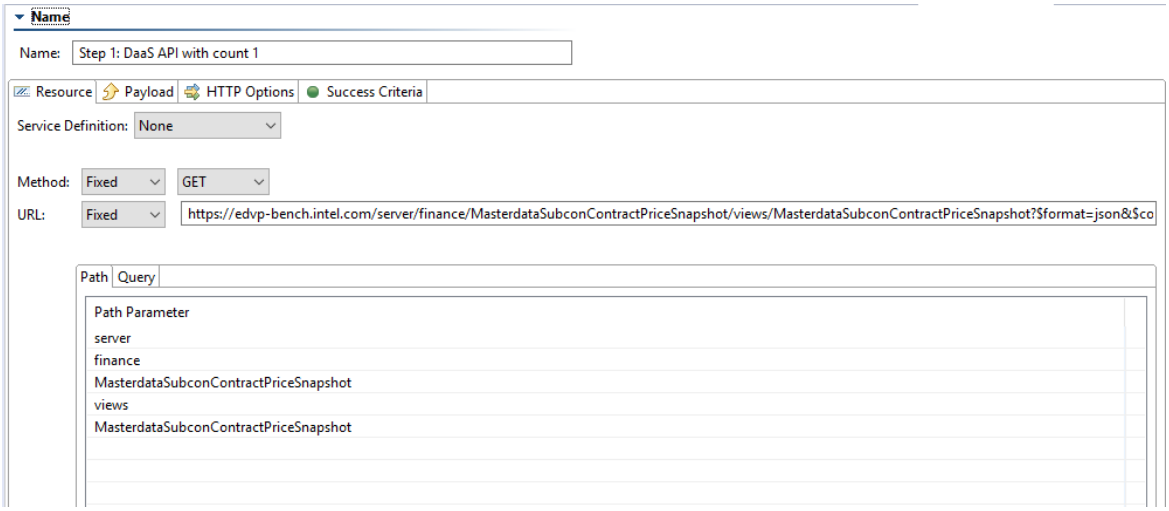


ILUSTRACIÓN 33 - INFORMACIÓN DE LA PRUEBA DE CLIENTE REST

3. Validación de la propuesta

En ICOST no se realizaban pruebas formales a nivel unitario y muy pocas a nivel de interfaz gráfica por lo que no se cuenta con números base para su medición. Las pruebas de regresión son parte de las pruebas que no realizaban anteriormente, por lo que no contamos con una base para medir la mejora en términos de tiempo ahorrado en ejecución sin embargo podemos observar una mejora en el producto de ICOST gracias estas pruebas automatizadas ya que manualmente no eran ejecutadas y ahora con la implementación automatizada pueden ser fácilmente ejecutadas para cubrir las pruebas de regresión en cada iteración o cada dos semanas de trabajo y cambios en la aplicación ICOST, además de la definición de un marco de trabajo establecido para la automatización de estas pruebas.

Por medio del marco de trabajo implementado, los analistas y desarrolladores de ICOST, crearon las pruebas de forma automatizada, como se explicó anteriormente, las pruebas actuales eran básicas, sin un orden, manuales y llevaban mucho tiempo, con la implementación del marco y gracias a las diferentes pruebas identificadas por medio de las reuniones y entrevistas realizada a lo largo de la elaboración del proyecto, los desarrolladores y analistas pudieron crear sus propias pruebas automatizadas utilizando Certify para las pruebas de interfaz de usuario.

De las pruebas automatizadas por los analistas y desarrolladores de ICOST se reportó que de forma manual la duración de la prueba es de aproximadamente dos horas para el área de Screens y de forma automatizada con Certify dura aproximadamente 5 minutos, por otro lado, las pruebas manuales del área de Reports, se reportó una duración de dos horas y media aproximadamente para cada prueba por medio de observación de los analistas, y de forma automatizada duran 6 minutos aproximadamente.

Áreas	Duración Manual Estimada	Duración Automatizada
Pantallas	2 horas	6 minutos
Reportes SSRS	2 horas y media	8 minutos
Procedimientos Almacenados/ Pruebas Unitarias	No hay registros	8 segundos

TABLA 7 - COMPARACIÓN DE TIEMPOS DE EJECUCIÓN DE PRUEBAS

Se hicieron pruebas con diferentes Screens y Reports para obtener un estimado de la duración que se daba debido a cada tipo de prueba, en total hay actualmente 60 Reports y 80 Screens por lo que si queremos revisar cuanto tiempo se ahorra por la automatización de dichas pruebas obtenemos la siguiente tabla:

Tipo de prueba	Duración Estimada	Cantidad de pruebas	Tiempo total
Pantallas Manual	120 minutos	80	9600 minutos/ 160 horas/ 6.67 días
Pantallas Automatizadas	6 min	80	480 minutos/ 8 horas/ 0.34 días
Reportes SSRS Manual	150 minutos	60	9000 minutos/ 150 horas/ 6.25 días
Reportes SSRS Automatizados	8 minutos	60	480 minutos/ 8 horas/ 0.34 días
Procedimientos Almacenados/ Pruebas Unitarias Automatizadas	0.14 minutos	1400 aproximadamente	196 minutos/ 3.26 horas/ 0.14 días

TABLA 8 – COMPARACIÓN DE TIEMPOS ENTRE PRUEBAS AUTOMATIZADAS Y MANUALES

Como se observa en la Tabla 8-Comparación de tiempos entre pruebas automatizadas y manuales, las diferencias de realizar las pruebas entre manual y automatizado en bastante alto el tiempo ahorrado. Tomando como ejemplo la duración de 6.67 días de las pruebas de Pantallas de forma manual se ve que es mucho tiempo el que se necesita, por lo que en ICOST se abstenían de realizar pruebas de regresión debido a la gran inversión de recursos necesario.

Con la cantidad de pruebas existentes actualmente, gracias a la automatización, en 1 día se podrían ejecutar todas, en el caso de que se llegue a automatizar 100%. Además de la gran mejora de tiempo en ejecución, otro punto a favor es que no se necesita un recurso humano 100% enfocado en realizar la pruebas, a lo que de ahora en adelante se podrán realizar las pruebas de regresión cuando sea necesario sin tener que sacrificar recursos y tiempo del proyecto ICOST.

Las pruebas unitarias de procedimientos almacenados que anteriormente no se estaba realizando, ahora que se estableció el marco de trabajo se pueden ejecutar cuando sea necesario y al estar automatizados con SOATest el tiempo necesario para ejecutar las pruebas es mínimo.

Todos los miembros requeridos para las pruebas de interfaz tienen y continuarán teniendo acceso a la herramienta Certify para poder crear nuevas pruebas identificadas a lo largo de la duración del proyecto ICOST, se dio la capacitación necesaria de cómo utilizar la herramienta y el marco de trabajo.

Como requisitos para la validación de estas pruebas de interfaz de tienen:

- Desarrolladores y analistas capacitados en la herramienta Certify y marco de trabajo implementado para esta
- Desarrolladores y analistas con acceso a Certify
- Desarrolladores y analistas con acceso a la página de ICOST junto con las áreas de Pantallas y Reportes SSRS
- Resultados de tiempos de ejecución de pruebas de forma manual.

Para las pruebas unitarias que son las pruebas que anteriormente no estaban ejecutando ni analizando en ICOST, se implementaron algunas pruebas básicas que se usaron como base para la creación del marco de trabajo para pruebas unitarias, de igual manera los desarrolladores ya hicieron uso del marco de trabajo creando así sus primeras pruebas unitarias.

Como se mencionó, las pruebas unitarias no se ejecutaban, por lo que no hay un registro de la duración actual de las pruebas, por lo que para los resultados de métricas vamos a basarnos en los datos de las pruebas automatizadas creadas con el marco de trabajo y la herramienta SOATest; la duración actual de las pruebas es de entre 10 a 15 segundos.

Para la validación de estas pruebas era necesario el cumplimiento de los siguientes requisitos:

- Desarrolladores capacitados en la herramienta SOATest y en el marco de trabajo creado para este
- Desarrolladores con acceso a la licencia y herramienta SOATest
- Desarrolladores con acceso al repositorio de GitLab
- Permisos de ingreso a la base de datos de ICOST por parte de los desarrolladores
- Pruebas unitarias identificadas para automatización

Con esta implementación se les garantiza a los líderes de ICOST un mejor seguimiento a las tareas de pruebas y de desarrollo, ya que, a la hora de un nuevo desarrollo o cambio en el código, se le ingresa al desarrollador una tarea de actualizar o crear el caso de prueba respectivo, el cual es creado y ejecutado utilizando el marco de trabajo de forma sencilla.

Los datos de la duración de la ejecución de las pruebas de interfaz y de unidad pueden tender a variar dependiendo del tamaño de la prueba, procesamiento de la base de datos, conexión a internet y otros factores externos a la herramienta o marco de trabajo.

Como otros puntos de validación se tiene el cumplimiento de los objetivos que se plantearon para la ejecución del proyecto.

Se realizó la investigación de varios marcos de trabajo de automatización de pruebas, la mayor parte de los artículos encontrados sobre marcos de trabajo o de automatización de pruebas dieron una ayuda o al menos nos brindaron una posible idea para la solución implementada en este proyecto para cumplir con el propósito de la creación de un marco de trabajo personalizado para el proyecto ICOST.

Algunos de los artículos revisados es el proyecto llamado “TITANIUM FRAMEWORK PARA AUTOMATIZACIÓN DE PRUEBAS DE SOFTWARE”, que a pesar de que la solución ejecutada en ese proyecto no utiliza las mismas herramientas que se han utilizado en nuestro proyecto, nos

brindó información e ideas sobre pruebas basadas en data driven y key driven, otra referencia revisada sobre marcos de trabajo es “Un marco de trabajo para el desarrollo y ejecución de pruebas automatizadas aplicadas al front-end de una aplicación web”, este proyecto nos impulsó a continuar con nuestra idea de completar nuestro proyecto utilizando una metodología ágil

Adicionalmente la usabilidad que brinda este marco de trabajo a la hora de desarrollar nuevas automatizaciones es de gran beneficio, los desarrolladores pueden crear nuevas automatizaciones a partir de ingresar nuevos datos en las bases de datos sin poseer conocimiento en las herramientas Certify ni SOATest debido que se establecieron todos los procesos genéricos y son dirigidos por los datos. Asimismo, el mantenimiento de las pruebas automatizadas se hace más factible ya que es en una única fuente de datos en donde se debe realizar cualquier modificación.

A continuación, se muestra el proceso realizado de manera gráfica con el fin de dar una mayor claridad al proceso en general de la definición del marco de trabajo para las pruebas automatizadas tanto a nivel de UI como unitarias. Se adjunta en el anexo 29 para una mejor resolución.

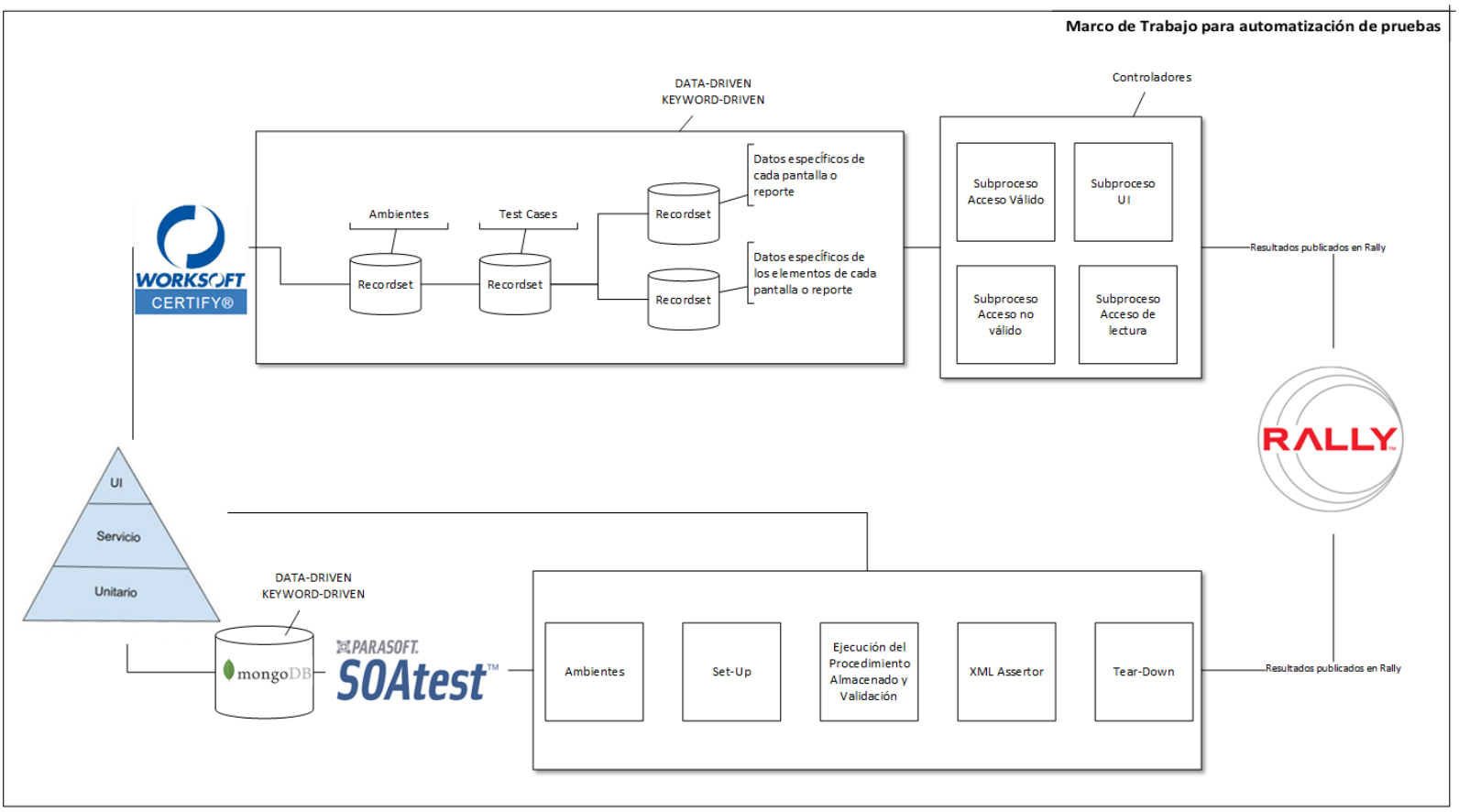


ILUSTRACIÓN 34 - GRÁFICO DEL MARCO DE TRABAJO

CAPÍTULO V
CONCLUSIONES Y RECOMENDACIONES

CAPÍTULO V: CONCLUSIONES Y RECOMENDACIONES

1. Conclusiones

Las conclusiones resumen los puntos más relevantes del proyecto, tomando en cuenta el análisis previo, la investigación y la propuesta de solución y los resultados del plan piloto.

1. Los objetivos propuestos para este proyecto fueron cumplidos en un 100%.
2. Un análisis previo a la implementación de la propuesta facilita la recolección de información necesaria para una solución óptima.
3. La técnica de grupo focal y entrevistas permiten que los colaboradores de un proyecto profundicen más sobre una problemática con el fin de generar una solución que beneficie el proyecto.
4. El analizar los tiempos de ejecución tanto manuales como automatizados permite garantizar una reducción de tiempo y obtener valor de las automatizaciones.
5. Generar un marco de trabajo basado en conceptos importantes como pruebas basadas en datos y pruebas basadas en palabras claves, además de incorporar las validaciones más frecuentes realizadas en el sistema ICOST.
6. La integración con la herramienta Rally para el manejo de las pruebas provee una trazabilidad y métricas sobre las automatizaciones.
7. La integración con la base de datos MongoDB permite mayor flexibilidad para el manejo de datos para controlar las pruebas automatizadas.
8. La integración con Gitlab permite el control de versiones de las pruebas automatizadas y la ejecución automática de las pruebas.
9. El plan piloto permitió probar el marco de trabajo a tiempo real con ejemplos reales del proyecto ICOST tanto a nivel de interfaz gráfica como de pruebas unitarias a nivel de código.
10. El éxito del plan piloto genera un impacto positivo en las actividades de pruebas en el proyecto ICOST tanto de Costa Rica como de Estados Unidos y Malasia, ya que lo incorporaron a sus procesos de desarrollo con el fin de mejorar la calidad del sistema.

2. Limitaciones

Como primera limitación que se enfrentó a lo largo del desarrollo del proyecto, es el tiempo, todos los involucrados en el desarrollo de este trabajo, cuentan con un empleo al cual deben cumplir, por lo tanto, no siempre estaban disponibles para consultas, o para la necesidad específica del momento, también afectan los asuntos personales de los involucrados, dicha limitación se solucionó por medio de buena organización y reuniones planeadas e informadas con anticipación a los involucrados.

El análisis de la situación actual, recolección general de datos por medio de reuniones, entrevistas y cuestionarios, también la elección de las herramientas y el análisis de cada una como mejor opción para el proyecto, aprender a usar las herramientas, influyeron como una limitante de tiempo, ya que era bastante trabajo y tiempo el que se le debía dedicar; como solución al igual que la limitante anterior de disponibilidad de personal, fue la organización y planeación adecuada.

Por otra parte, también podemos tomar como parte de limitante de tiempo son los accesos a las diferentes instancias del proyecto ICOST, pues para obtener accesos se debían solicitar por medio de la herramienta de Intel llamada AGS, además de que no solo por parte de ICOST eran necesarios los accesos, sino también para el uso de las herramientas de automatización Certify y SOATest, y se duraban varios días para completar su aprobación.

Durante el transcurso del proyecto hubo varios cambios en el personal de ICOST, específicamente en los facilitadores del proyecto (scrum masters) y los dueños del producto (product owners), entre los cuales se encontraba el dueño del producto que habilitó realizar el proyecto en ICOST por lo que hubo que conversar con el nuevo encargado para explicar el proyecto que se estaba efectuando y llegar nuevamente a un acuerdo sobre el proyecto.

3. Trabajos futuros o recomendaciones

1. Se recomienda a los líderes y colaboradores de ICOST implementar a cabalidad las pruebas automatizadas utilizando el marco de trabajo implementado con el fin de que se optimicen los procesos y obtengan todos los beneficios.
2. Los colaboradores de ICOST deben mantener actualizadas las pruebas automatizadas para seguir obteniendo retorno de inversión (ROI) como mínimo cada 6 meses ejecutar todas las pruebas y validar el correcto funcionamiento.
3. Se recomienda capacitar a nuevos colaboradores que ingresen a ICOST en las herramientas utilizadas como SOATest y Certify.
4. Es importante que los líderes sigan apoyando esta iniciativa con el fin de obtener mayores beneficios a lo largo del tiempo debido que entre más pruebas se logren automatizar mayor es el ROI que se obtiene y mejor es la calidad del sistema.
5. Se recomienda a los colaboradores de ICOST continuar con sesiones de grupos focales con el fin de identificar y determinar nuevas áreas y nuevas funcionales para agregar al marco de trabajo con el fin de tener una mayor cobertura.
6. Se recomienda continuar investigando sobre herramientas de automatización o integrar con nuevas herramientas que se utilicen en Intel como parte del manejo de pruebas.
7. Se recomienda ampliar el alcance de la ejecución de pruebas automatizadas, ya sea utilizar un calendario de ejecución de pruebas o integrar Gitlab para las pruebas de interfaz gráfica, los colaboradores de ICOST pueden trabajar en conjunto con el equipo de herramientas de Intel con el fin de obtener guía para dicha implementación y además para obtener máquinas virtuales adicionales para estas

ejecuciones. Al colocar todas las pruebas en ejecución automática les permite estar probando constantemente el sistema y obtener los resultados en un único lugar como lo es Rally utilizando la integración realizada en este proyecto.

8. Se recomienda, si el presupuesto del departamento lo permite, tener un colaborador dedicado 100% a la calidad del sistema y a la automatización de pruebas con el fin de encontrar mejoras y nuevas tecnologías.

REFERENCIAS

REFERENCIAS

Gutiérrez, J. (s. f.). *¿Qué es un framework web?* http://www.lsi.us.es. http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf

Minnetto, E. (2007). Frameworks para desenvolvimiento en PHP, Brasil: Novatec.

Kaur, M., & Kumari, R. (2011). Comparative Study of Automated Testing Tools: TestComplete and QuickTest Pro. *International Journal of Computer Applications*, *24*(1), 1-7. https://doi.org/10.5120/2918-3844

Giménez, M., & Espínola, A. (2014). *Automatizacion de Pruebas para Interfaces de ´ Aplicaciones Web*. http://www.cc.pol.una.py/wpfg2014/

Laukkanen, P. (2006). *Data-Driven and Keyword-Driven Test Automation Frameworks* (Maestría). HELSINKI UNIVERSITY OF TECHNOLOGY.

Jain, A., & Sharma, S. (2012). AN EFFICIENT KEYWORD DRIVEN TEST AUTOMATION FRAMEWORK FOR WEB APPLICATIONS. *AN EFFICIENT KEYWORD DRIVEN TEST AUTOMATION FRAMEWORK FOR WEB APPLICATIONS*, *2*(3), 600-604. https://www.ijesat.org/

Mascheroni, M., Cogliolo, M., & Irrazabal, E. (2016). Automatización de pruebas de compatibilidad web en un entorno de desarrollo continuo de software. *Simposio Argentino de Ingeniería en Software*, *17*, 51-63. http://sedici.unlp.edu.ar/

Díaz, M. (2017). *Sistema para el Control de Traslados Telefónicos Pendientes* (Diplomado). Universidad Central “Marta Abreu” de Las Villas.

A, J. (2020, 1 septiembre). *What is Component Testing Or Module Testing (Learn With Examples)*. Software Testing Help. https://www.softwaretestinghelp.com/what-is-component-testing-or-module-testing/

Ramos, J. (s. f.). *Los diferentes tipos de Pruebas de software*. https://programacionymas.com/. https://programacionymas.com/blog/tipos-de-testing-en-desarrollo-de-software

B. (2020, 4 agosto). *Best Automation Testing Tools for 2020 (Top 10 reviews)*. Medium. https://medium.com/@briananderson2209/best-automation-testing-tools-for-2018-top-10-reviews-8a4a19f664d2

Atlassian. (s. f.). *The different types of testing in Software*. https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing

FEWSTER, M., & GRAHAM, D. (1999). Software Test Automation Effective use of test execution tools. Addison-Wesley.

Hooda, I.., & Singh Chhillar, R.. (2015). Software Test Process, Testing Types and Techniques. International Journal of Computer Applications, 111(13), 11. https://www.ijcaonline.org/

Díaz, J., Banchoff Tzancoff, C., Rodríguez, A., & Soria, V. (2009, agosto). Herramientas open source para testing de aplicaciones Web. Evaluación y usos. Sedici UNPL. http://sedici.unlp.edu.ar/handle/10915/21017

Rodriguez, M. L. (19 de Noviembre de 2010). metodologiasdelainvestigacion.wordpress.com. Obtenido de https://metodologiasdelainvestigacion.wordpress.com/2010/11/19/la-tecnica-de-la-encuesta/

Silveira Donaduzzi, D. S. . d. a., Colomé Beck, C. L., Heck Weiller, T., Nunes da Silva Fernandes, M., & Viero, V. (2015). Grupo focal y análisis de contenido en investigación cualitativa. *Index de Enfermería*, *24*(1-2), 71-75. https://doi.org/10.4321/s1132-12962015000100016

Zavaleta, J. (2016). Los grupos focales como estrategia para recolectar información. Obtenido de http://www.espolea.org/uploads/8/7/2/7/8727772/ddt-gruposfocales.pdf

José Amícola, El poder-femme. Virginia Woolf, Simone de Beauvoir y Victoria Ocampo. La Plata, EDULP, 2019, Género, 263 páginas. Edición digital disponible en: http://sedici.unlp.edu.ar/handle/10915/80598 (Vol. 25, Número 31). (2020). Universidad Nacional de La Plata. https://doi.org/10.24215/18517811e156

Echeverría Perez, D., & Paumier, A. A. (2014). Testing como Práctica para Evaluar la Eficiencia en Aplicaciones Web. *Revista Latinoamericana de Ingenieria de Software*, *2*(5), 307-309. https://doi.org/10.18294/relais.2014.307-309

Campos, G., & Lule, N. (2012). LA OBSERVACIÓN, UN MÉTODO PARA EL ESTUDIO DE LA REALIDAD. Xihmai, 7, 45-60. https://dialnet.unirioja.es/

Sánchez, G. (2016). TITANIUM FRAMEWORK PARA AUTOMATIZACIÓN DE PRUEBAS DE SOFTWARE. *Pistas Educativas, 40*, 1027-1043. http://itcelaya.edu.mx/

Fernández, S. (2016). *Un marco de trabajo para el desarrollo y ejecución de pruebas automatizadas aplicadas al front-end de una aplicación web* (Grado en Ingeniería Informática). Universitat Politècnica de València.

Procedimientos almacenados (motor de base de datos) - SQL Server. (2017, 14 marzo). Microsoft Docs. https://docs.microsoft.com/es-es/sql/relational-databases/stored-procedures/stored-procedures-database-engine?view=sql-server-ver15

Glosario de Términos

ICOST: Se refiere a una aplicación desarrollada y utilizada por Intel Corporation para la gestión del costeo y rotación de inventarios a nivel de manufactura. Adicionalmente, se nombra ICOST el equipo que soporta esa aplicación.

Framework o marco de trabajo: es una estructura definida para resolver un problema

RPA: Robotic Process Automation se refiere a la tecnología de desarrollar robots virtuales los cuales reproducen acciones humanas en los

PO: Product Owner se refiere al dueño de la aplicación el cual define y valida los requerimientos del proyecto

SM: Scrum master se refiere a la persona encargada de dirigir las reuniones o ceremonias establecidas en Agile.

APT: Agile persistent team se refiere al equipo de trabajo para realizar de manera ágil un desarrollo de un proyecto siguiendo los lineamientos de Agile.

DSU: Daily stand up es una reunión diaria del manifiesto ágil con el fin de que los miembros del equipo colaboren entre sí y aumenten su productividad.

GUI: Graphical user interface o interfaz de usuario son programas informáticos con la funcionalidad de interfaz de usuario por medio de imágenes u objetos para representar la información o acciones disponibles.

API: Application Programming Interface o interfaz de programación de aplicaciones, ofrece funcionalidades y procedimientos de ciertas bibliotecas que pueden ser utilizados por otras aplicaciones.

Controllers: O controladores, en el caso de Certify los controladores nos permiten agrupar procesos para ser ejecutados de forma conjunta.

UI: Interfaz de usuario, es lo que el usuario puede ver y utilizar por medio del software.

Stored procedures: Procedimientos Almacenados, es un grupo de instrucciones que son ejecutadas por el motor de base de datos, pueden tener parámetros de entrada, lógica, pueden retornar un valor o ejecutar alguna función interna en la base de datos

ANEXOS

ANEXOS

Anexo 1 – Plantilla de minuta de reunión

Fecha		**Hora inicio**	
Lugar		**Hora fin**	
Objetivo			

Asistentes

Asistentes	
Nombre	**Puesto**

Asuntos tratados

Compromisos asumidos

No.	**Tarea**	**Responsable**
1		

Anexo 2 - Minuta de reunión #1

Fecha	05 noviembre, 2018	**Hora inicio**	3:00
Lugar	Intel	**Hora fin**	4:00
Objetivo			
Entender el proceso de pruebas manuales a nivel de interfaz en la aplicación ICOST.			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Scott McDonald	Ingeniero en Calidad de Software

Asuntos tratados

Proceso actual de manejo de pruebas manuales.

Niveles de testing actualmente implementados en ICOST.

Contactos para las diversas áreas de ICOST para una comunicación directa.

Componentes de la aplicación ICOST como lo son las pantallas y reportes SSRS.

Compromisos asumidos

No.	Tarea	Responsable
1	Contactar las diferentes personas de las diferentes áreas de ICOST para obtener una mejor visión de la aplicación y entender el funcionamiento.	Juan Diego González Arias y Mariela Sequeira Ugalde
2	Explorar la herramienta Certify e identificar qué niveles de testing se pueden implementar	Juan Diego González Arias y Mariela Sequeira Ugalde
3	Entender el funcionamiento de los reportes SSRS y pantallas de ICOST.	Juan Diego González Arias y Mariela Sequeira Ugalde
4	Priorizar las funcionalidades de los reportes SSRS y pantallas de ICOST	Scott McDonald

Anexo 3 - Minuta de reunión #2

Fecha	12 diciembre, 2018	**Hora inicio**	2:00
Lugar	Intel	**Hora fin**	2:30
Objetivo			
Conocer las opciones disponibles de buenas prácticas y automatización.			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Nazia Khan	Ingeniero en Desarrollo de Software

Asuntos tratados

Mejores prácticas de calidad y automatización de Intel IT.

Tipos de pruebas que se deberían automatizar.

Herramientas de automatización disponibles en Intel IT.

Compromisos asumidos

No.	Tarea	Responsable
1	Análisis del proceso actual de pruebas en ICOST, detalles y posibles defectos.	Juan Diego González Arias y Mariela Sequeira Ugalde
2	Tipos de pruebas automatizadas que se deberían implementar basados en la tarea 1.	Juan Diego González Arias y Mariela Sequeira Ugalde
3	Elección de herramientas disponibles en Intel basados en la tarea 2.	Juan Diego González Arias y Mariela Sequeira Ugalde

Anexo 4 - Minuta de reunión #3

Fecha	21 enero, 2019	**Hora inicio**	10:00
Lugar	Intel	**Hora fin**	10:30
Objetivo			
Conocer directamente de la patrocinadora del proyecto los diferentes puntos críticos del proyecto que vamos a tratar			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Jacqueline Miranda	Dueña del producto y patrocinadora

Asuntos tratados

Problemática actual con respecto a las pruebas de software y puntos a mejorar.

Compromisos asumidos

No.	Tarea	Responsable
1	Análisis personal sobre la problemática actual	Juan Diego González Arias y Mariela Sequeira

Anexo 5 - Minuta de reunión #4

Fecha	19 febrero, 2019	**Hora inicio**	9:00
Lugar	Intel	**Hora fin**	9:30
Objetivo			
Entender más a fondo la problemática y además exponer la posible solución			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Levi Chang	Jefe de Producto ICOST (Chief Product Owner)

Asuntos tratados

Situación actual de ICOST, problemáticas, desafíos, nuevas actualizaciones y funcionalidades.

Plan de trabajo del proyecto a alto nivel.

Frecuencia de los procesos de pruebas en el proyecto.

Visión general de la propuesta solución.

Compromisos asumidos

No.	Tarea	Responsable
1	Presentar avance de la propuesta solución junto con el plan de trabajo.	Juan Diego González Arias y Mariela Sequeira Ugalde
2	Estimar tiempo a ahorrar y beneficios a obtener al automatizar los procesos.	Juan Diego González Arias y Mariela Sequeira Ugalde

Anexo 6 - Minuta de reunión #5

Fecha	08 marzo, 2019	**Hora inicio**	11:00
Lugar	Intel	**Hora fin**	12:00
Objetivo			
Conocer más a fondo la herramienta Certify que fue la seleccionada para realizar la automatización de las pruebas de interfaz de usuario			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
John Sullivan	Ingeniero en Desarrollo de Software/Experto en Certify

Asuntos tratados

Características de la herramienta, funcionalidades, información de cómo se utiliza y permisos necesarios para instalar y hacer uso de esta.

Compromisos asumidos

No.	Tarea	Responsable
1	Solicitar permisos de acceso al uso de Certify por medio de la herramienta de accesos de Intel	Juan Diego González Arias y Mariela Sequeira
2	Realizar pruebas utilizando Certify y generar dudas para ser aclaradas por el experto	Juan Diego González Arias y Mariela Sequeira

Anexo 7 - Minuta de reunión #6

Fecha	12 abril, 2019	**Hora inicio**	2:00
Lugar	Intel	**Hora fin**	2:30
Objetivo			
Recopilar información desde la perspectiva del facilitador del proyecto, además para conversar sobre la posible solución.			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Juan Carlos Murillo	Facilitador del proyecto ICOST (Scrum Master)

Asuntos tratados

Desafíos a la hora de realizar procesos de pruebas manuales.

Presentación de la solución propuesta.

Retroalimentación a partir de la propuesta como lo es incorporar dentro de las funcionalidades a probar de las pantallas, las pruebas de datos históricos.

Compromisos asumidos

No.	Tarea	Responsable
1	Incorporar a la propuesta la funcionalidad de validación de históricos.	Juan Diego González Arias y Mariela Sequeira Ugalde

Anexo 8 - Minuta de reunión #7

Fecha	06 junio, 2019	**Hora inicio**	11:00
Lugar	Intel	**Hora fin**	11:30
Objetivo			
Obtener retroalimentación de la solución propuesta a un nivel más alto de jefatura.			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Michael Weir	Jefatura de Producto ICOST

Asuntos tratados

Presentación de solución propuesta del marco de trabajo con las diversas herramientas Certify y SOATest.

Presentación de métodos como Data-Driven y KeyWord-Driven para un mayor beneficio de las automatizaciones.

Estimación de beneficios.

Retroalimentación como lo es enfocarnos en áreas de mayor volumen de usuarios.

Compromisos asumidos

No.	Tarea	Responsable
1	Enfocar las pruebas a incorporar en la solución en base a las áreas de prioridad.	Juan Diego González Arias y Mariela Sequeira Ugalde

Anexo 9 - Minuta de reunión #8

Fecha	07 agosto, 2019	**Hora inicio**	4:00
Lugar	Intel	**Hora fin**	5:00
Objetivo			
Conocer los procesos de SQL más importantes y entender las funcionalidades a probar.			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Jorge Grimaldi	Ingeniero en Desarrollo de Software

Asuntos tratados

Revisión de procesos actuales que utilizan la herramienta SQL Server y proceso actual de prueba de estos, accesos para hacer uso de la base de datos de ICOST.

Compromisos asumidos

No.	Tarea	Responsable
1	Solicitar permisos de acceso a la base de datos de ICOST por medio de la herramienta de accesos de Intel	Juan Diego González Arias y Mariela Sequeira
2	Realizar pruebas utilizando lenguaje SQL en la base de datos ICOST	Juan Diego González Arias y Mariela Sequeira

Anexo 10 - Minuta de reunión #9

Fecha	16 octubre, 2019	**Hora inicio**	2:00
Lugar	Intel	**Hora fin**	2:30
Objetivo			
Entender desde el punto de vista de un desarrollador y entender posibles escenarios faltantes.			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Srikanth Bongu	Ingeniero en Desarrollo de Software

Asuntos tratados

A partir de un área de prioridad de ICOST por su volumen de usuarios, se revisó las funcionalidades cubiertas. Se determinó que era importante incorporar la validación de la funcionalidad de exportar los datos a Excel.

Así mismo se identificaron los procedimientos almacenados prioritarios para las pruebas a agregar a la solución.

Compromisos asumidos

No.	Tarea	Responsable
1	Incorporar funcionalidad de validación de exportar a Excel al marco de trabajo en Certify	Juan Diego González Arias y Mariela Sequeira Ugalde
2	Revisar los procedimientos almacenados y crear las pruebas necesarias y de ser necesario agendar otra reunión para una mayor comprensión de estos.	Juan Diego González Arias y Mariela Sequeira Ugalde

Anexo 11 - Minuta de reunión #10

Fecha	11 noviembre, 2019	**Hora inicio**	4:00
Lugar	Intel	**Hora fin**	4:30
Objetivo			
Obtener retroalimentación del trabajo realizado al momento.			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Juan Carlos Murillo	Ingeniero en Desarrollo de Software/Experto en SCRUM

Asuntos tratados

Revisión del progreso actual con respecto a la automatización de los procesos analizados.

Anexo 12 - Minuta de reunión #11

Fecha	18 febrero, 2020	**Hora inicio**	2:00
Lugar	Intel	**Hora fin**	3:00
Objetivo			
Entender la herramienta SOATest, las funcionalidades y limitaciones que posee, los entrenamientos disponibles, el soporte de la herramienta, entre otros aspectos.			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Jesse Fitterer	Ingeniero en Desarrollo de Software

Asuntos tratados

Explicación y demo de la herramienta SOATest por parte de Jesse.

El soporte de la herramienta SOATest es brindado por el equipo de Herramientas de IT de Intel.

Ellos mismos brindan entrenamientos a empleados de Intel que lo requieran.

La herramienta SOATest es adecuada para realizar las pruebas automatizadas de procedimientos almacenados de SQL.

Compromisos asumidos

No.	Tarea	Responsable
1	Compromiso con los estudiantes para un acompañamiento durante el proceso de desarrollo, accesos y entrenamiento de la herramienta SOATest.	Jesse Fitterer
2	Tomar entrenamientos necesarios de la herramienta SOATest	Juan Diego González Arias y Mariela Sequeira Ugalde
3	Agendar reuniones con Jesse Fitterer para seguimientos.	Juan Diego González Arias y Mariela Sequeira Ugalde

Anexo 13 - Minuta de reunión #12

Fecha	07 mayo, 2020	**Hora inicio**	4:00
Lugar	Intel	**Hora fin**	4:30
Objetivo			
Obtener retroalimentación sobre trabajo realizado hasta el momento, enfocado en las pruebas unitarias en SQL.			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Jorge Grimaldi	Ingeniero en Desarrollo de Software

Asuntos tratados

Revisión del progreso actual enfocado en las pruebas unitarias de SQL, opinión y posibles mejoras en el proceso automatizado por la herramienta SOA Test

Compromisos asumidos

No.	Tarea	Responsable
1	Ejecutar mejoras de las pruebas automatizadas utilizando SOATest	Juan Diego González Arias y Mariela Sequeira

Anexo 14 - Minuta de reunión #13

Fecha	23 junio, 2020	**Hora inicio**	2:00
Lugar	Intel	**Hora fin**	3:00
Objetivo			
Obtener retroalimentación sobre el trabajo realizado hasta el momento.			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Soledad Riggioni	Dueña de producto ICOST

Asuntos tratados

Revisión general del proyecto, tanto pruebas unitarias como de interfaz, opiniones y posibles mejoras.

Compromisos asumidos

No.	**Tarea**	**Responsable**
1	Ejecutar mejoras obtenidas por la retroalimentación en las pruebas automatizadas requeridas	Juan Diego González Arias y Mariela Sequeira

Anexo 15 - Minuta de reunión #14

Fecha	07 enero, 2019	**Hora inicio**	09:00
Lugar	Intel	**Hora fin**	09:30
Objetivo			
Obtener retroalimentación de la solución propuesta específicamente en Certify para las pruebas de regresión con el fin de obtener críticas constructivas.			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Scott McDonald	Ingeniero en Calidad de Software
Levi Chang	Dueño del producto

Asuntos tratados

Revisión de pruebas automatizadas utilizando Certify (pruebas de interfaz)

Compromisos asumidos

No.	Tarea	Responsable
1	Mejora de pruebas actuales e implementación de la retroalimentación para futuras pruebas	Juan Diego González Arias y Mariela Sequeira

Anexo 16 - Minuta de reunión #15

Fecha	07 marzo, 2019	**Hora inicio**	01:00
Lugar	Intel	**Hora fin**	01:30
Objetivo			
Reunión con diversos ingenieros de automatización de diversos proyectos dentro de IT, con el fin de conocer más opciones y herramientas y con el fin de obtener nuevas ideas.			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Heizel Perez	Ingeniero en Desarrollo de Software
Andrés Rodríguez	Ingeniero en Desarrollo de Software
Joseph Campos	Ingeniero en Desarrollo de Software

Asuntos tratados

Nuevas ideas sobre herramientas que se pueden utilizar para mejorar el proceso de pruebas. Revisión de algunas pruebas automatizadas actualmente.

Anexo 17 - Minuta de reunión #16

Fecha	10 mayo, 2019	**Hora inicio**	04:00
Lugar	Intel	**Hora fin**	04:30
Objetivo			
Reunión con analistas del sistema ICOST con el fin de revisar los escenarios de pruebas.			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Soledad Riggioni	
Zaily Rojas	Ingeniero en Desarrollo de Software/Analista de sistema
Adrián Campos	Ingeniero en Desarrollo de Software/ Analista de sistema

Asuntos tratados

Análisis de diferentes escenarios de pruebas, tanto de interfaz de usuario (Página web de ICOST, revisión de reportes y permisos de usuario) y pruebas de unidad (llamadas SQL)

Compromisos asumidos

No.	Tarea	Responsable
1	Automatizar con SOATest y Certify los escenarios de pruebas identificado en la reunión	Juan Diego González Arias y Mariela Sequeira

Anexo 18 - Minuta de reunión #17

Fecha	26 agosto, 2019	**Hora inicio**	03:30
Lugar	Intel	**Hora fin**	04:00
Objetivo			
Se realizó una reunión con los líderes para obtener retroalimentación de la solución propuesta además de mostrar el nuevo proceso automatizado para las pruebas de regresión en específico con el fin de obtener críticas constructivas.			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Juan Carlos Murillo	Facilitador del proyecto ICOST (Scrum Master)

Asuntos tratados

Mostrar las pruebas automatizadas y obtener retroalimentación.

Mejoras al proceso automatizado actual.

Anexo 19 - Minuta de reunión #18

Fecha	17 octubre, 2019	**Hora inicio**	11:30
Lugar	Intel	**Hora fin**	12:00
Objetivo			
Se realizó una reunión con líderes y personas clave para conversar sobre el proceso de regresión de pruebas en el proyecto ICOST y cómo la automatización beneficia dicho proceso.			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Jacqueline Miranda	Dueña del proyecto ICOST
Jean Marc Lenc	Líder del proyecto ICOST
Carlos Ovares	Ingeniero en Desarrollo de Software
Mauricio Otarola	Gerente de desarrolladores de ICOST

Asuntos tratados

Pruebas de regresión en ICOST

Beneficios principales de la automatización de dichas pruebas

Compromisos asumidos

No.	Tarea	Responsable
1	Realizar un demo para los líderes de las pruebas de interfaz	Juan Diego González y Mariela Sequeira

Anexo 20 - Minuta de reunión #19

Fecha	28 octubre, 2019	**Hora inicio**	2:00
Lugar	Intel	**Hora fin**	3:00
Objetivo			
Se obtuvo retroalimentación de parte de los desarrolladores específicamente de la solución desarrollada en la herramienta SOATest			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Jorge Grimaldi	Ingeniero en Desarrollo de Software
Daniela Castro	Ingeniero en Desarrollo de Software
Eduardo Rojas	Ingeniero en Desarrollo de Software
Katherine Corrales	Ingeniero en Desarrollo de Software

Asuntos tratados

Demo de la herramienta para la automatización de pruebas unitarias.

Anexo 21 - Minuta de reunión #20

Fecha	28 octubre, 2019	**Hora inicio**	2:00
Lugar	Intel	**Hora fin**	3:00
Objetivo			
Se obtuvo retroalimentación de parte de los desarrolladores específicamente de la solución desarrollada en la herramienta SOATest			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Jorge Grimaldi	Ingeniero en Desarrollo de Software
Daniela Castro	Ingeniero en Desarrollo de Software
Eduardo Rojas	Ingeniero en Desarrollo de Software
Katherine Corrales	Ingeniero en Desarrollo de Software

Asuntos tratados

Demo de la herramienta para la automatización de pruebas unitarias.

Anexo 22 - Minuta de reunión #21

Fecha	14 enero, 2020	**Hora inicio**	2:00
Lugar	Intel	**Hora fin**	3:00
Objetivo			
Reunión realizada con el fin de reunir expertos de ambas herramientas utilizadas en el proyecto, con el fin de ver la solución que se estaba desarrollando tanto como en Certify como en SOATest, además para obtener retroalimentación, guía, ideas nuevas, generar crítica constructiva y consejos que pudiéramos aplicar con las tecnologías que poseemos.			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Jesse Fitterer	Experto en SOATest
John Sullivan	Experto en Certify

Asuntos tratados

Demo de lo desarrollado hasta el momento.

Se obtuvo retroalimentación.

Se generaron ideas como la integración de la base de datos MongoDB y además la integración con rally para la publicación de resultados.

Compromisos asumidos

No.	Tarea	Responsable
1	Creación de la base de datos en MongoDB.	Juan Diego González y Mariela Sequeira
2	Entender el proceso de publicación de resultados en Rally.	Juan Diego González y Mariela Sequeira
3	Implementar la integración con Rally en el marco de trabajo realizado.	Juan Diego González y Mariela Sequeira

Anexo 23 - Minuta de reunión #22

Fecha	19 febrero, 2020	**Hora inicio**	2:00
Lugar	Intel	**Hora fin**	3:00
Objetivo			
Se desarrolló el tema de Data-driven específicamente para la herramienta SOATest con los desarrolladores de ICOST.			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Jorge Grimaldi	Ingeniero en Desarrollo de Software
Daniela Castro	Ingeniero en Desarrollo de Software
Eduardo Rojas	Ingeniero en Desarrollo de Software
Katherine Corrales	Ingeniero en Desarrollo de Software

Asuntos tratados

Demo de la solución específicamente de la base de datos de MongoDB a los desarrolladores.

Entrenamiento hacia los desarrolladores con el fin de una mayor adaptación en el proceso de desarrollo.

Anexo 24 - Minuta de reunión #23

Fecha	01 abril, 2020	**Hora inicio**	3:00
Lugar	Intel	**Hora fin**	4:00
Objetivo			
Obtener retroalimentación por parte de los desarrolladores sobre la solución implementada.			

Asistentes

Asistentes	
Nombre	**Puesto**
Juan Diego González Arias	Ingeniero en Desarrollo de Software
Mariela Sequeira Ugalde	Ingeniero en Desarrollo de Software
Jorge Grimaldi	Ingeniero en Desarrollo de Software
Daniela Castro	Ingeniero en Desarrollo de Software
Eduardo Rojas	Ingeniero en Desarrollo de Software
Katherine Corrales	Ingeniero en Desarrollo de Software

Asuntos tratados

Mostrar los resultados actuales con respecto a los procesos automatizado a los desarrolladores.

Cómo podríamos mejorarlo basados en la visión de ellos.

Anexo 25 - Manual técnico de SOATest: implementación

Este anexo se encuentra en idioma inglés debido a que es dirigido a la población del departamento de ICOST el cual está conformado por personas en Costa Rica, Estados Unidos y Malasia.

INSTALL SOATEST IN SERVER AND SETUP CI/CD USING GITLAB

Juan Diego González Arias, Mariela Sequeira Ugalde

Contents

1 Version History

Ver. #	Date	Updated by	Comments
Draft	25/11/2019	Mariela Sequeira Ugalde Juan Diego González Arias	First Draft Created

2 Scope

This document covers the installation steps to set up SOATest tool in a server and the configuration required in Gitlab to setup CI/CD.

3 Target Audience

The target audience of this document is DBA or whoever are responsible to setup this CI/CD process using SOATest and GITLAB.

4 Prerequisites

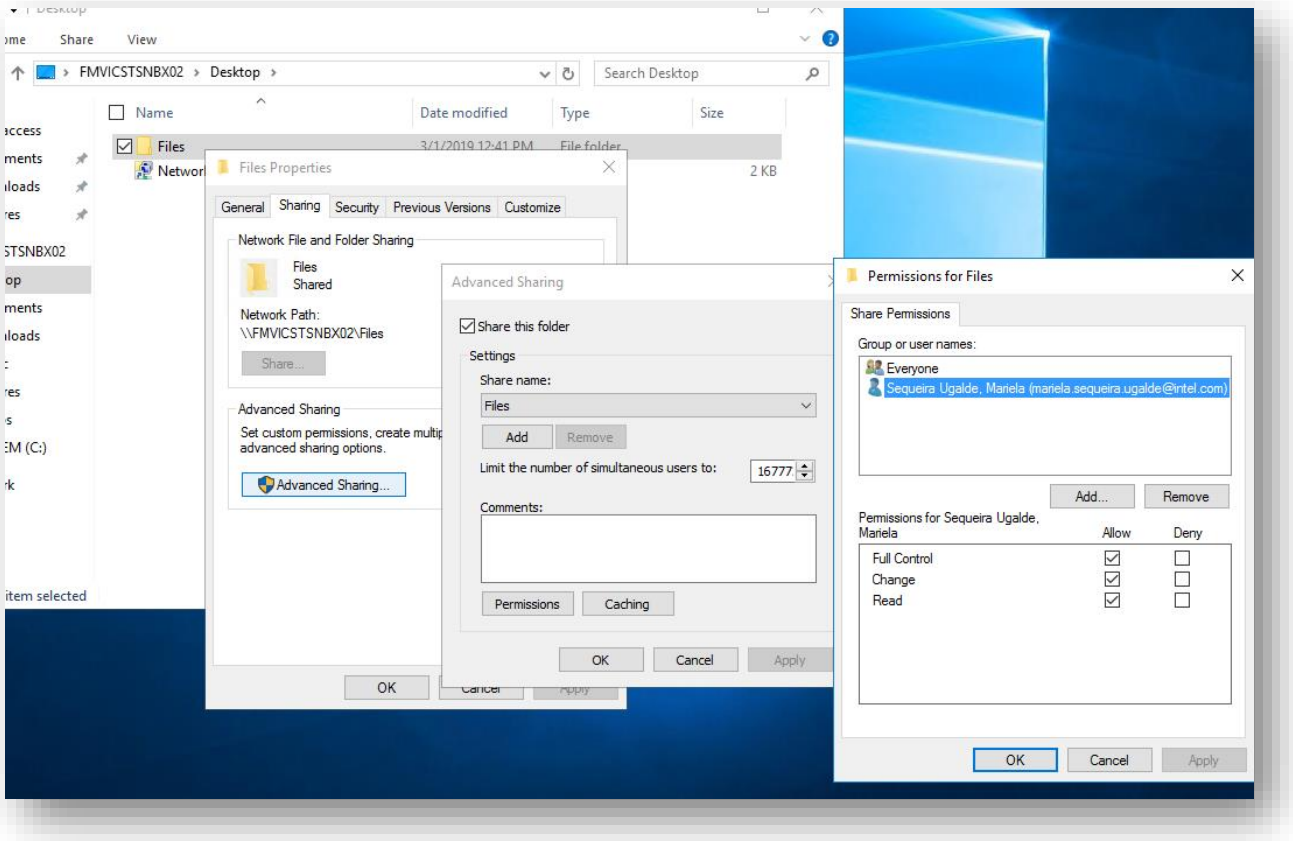
- You will need a Windows Server available and have administrator access to it.
- You will need to have access to Gitlab project so that you can clone the project and check in changes.
- You will need some knowledge of how to use SOATest and Gitlab.
    - This assumes you have already been approved for the "SOATest License Read Only License" Entitlement in AGS. Training must be completed prior to downloading the software. Here you can find training link https://intel.sabacloud.com/Saba/Web_spf/NA2PRD0003/common/searchresults/soatest/ALL
    - This assumes that you have Gitlab access to the specific project.
- You will need Tortoise or any other GIT tool.
- You will need enough drive space and RAM in your work environment.

5 Install SOATest in Server

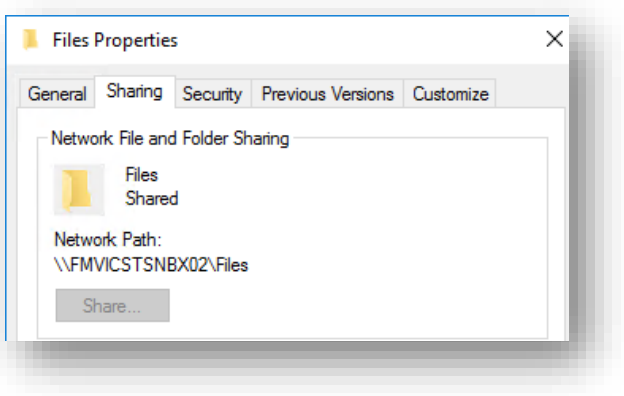
5.1 Set-Up Steps

5.1.1 Install and Set up SOATest

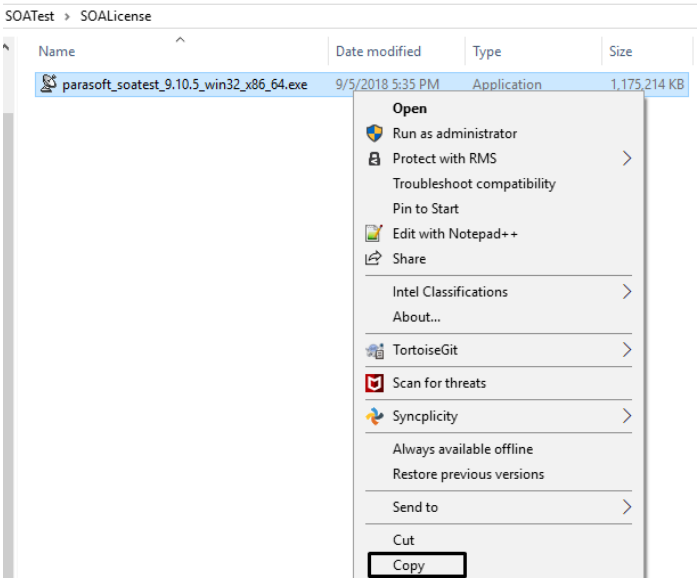
- Log in to the server, you can use remote desktop connection as an option.
- Create a folder (in desktop) with any name. This folder will help us to copy the executables of the programs that we need in server.
- Share this folder to you so you can copy the programs from your computer.
    - Right click Properties, sharing tab, advance settings, Click permissions.
    - Click add and add your idsid, give full access and click OK.



- Copy the network Path created above. And open it from your local computer. In this case \\FMVICSTSNBX02\Files



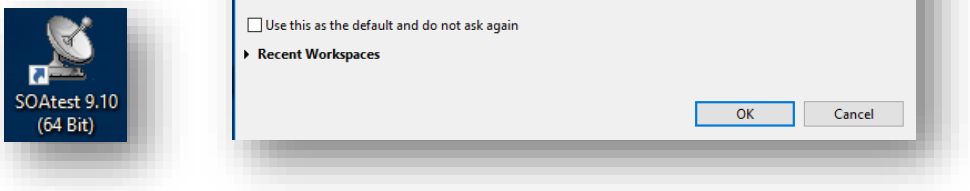
- Open a new file explorer and navigate to
    - AMR → \\10.2.67.165\SOATest\SOALicense
    - GAR → \\10.109.66.4\SOATest\SOALicense
    - Or ask mariela.sequeira.ugalde@intel.com to provide you the SOATest installer

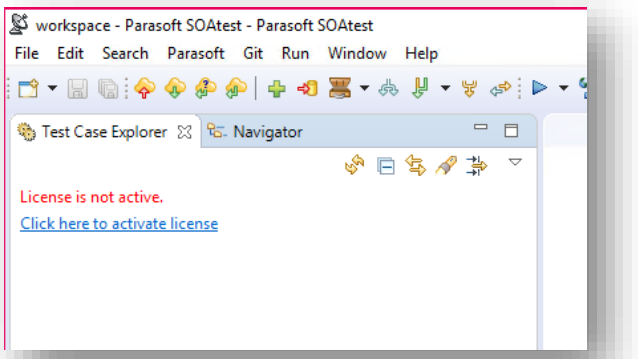


- Paste the file in the shared folder from server
- In the server you have the installer.
    - Run the executable as an administrator. (Right Click on the File)
    - Accept license terms and follow the install wizard instructions to install SOAtest.

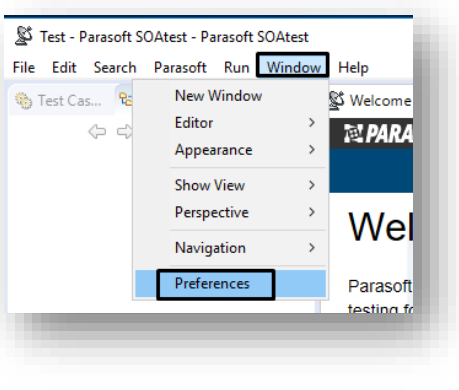
- Make sure checkbox is checked to create shortcut on your desktop to launch. Use all the defaults.

- Launch SOATest in the Server
  - Double Click on the SOATest Icon on your desktop
  - Input the workspace: C:\ICOST\workspace
  - Click Ok
  - If you receive the message “License is not active”. Click the link to activate the license.

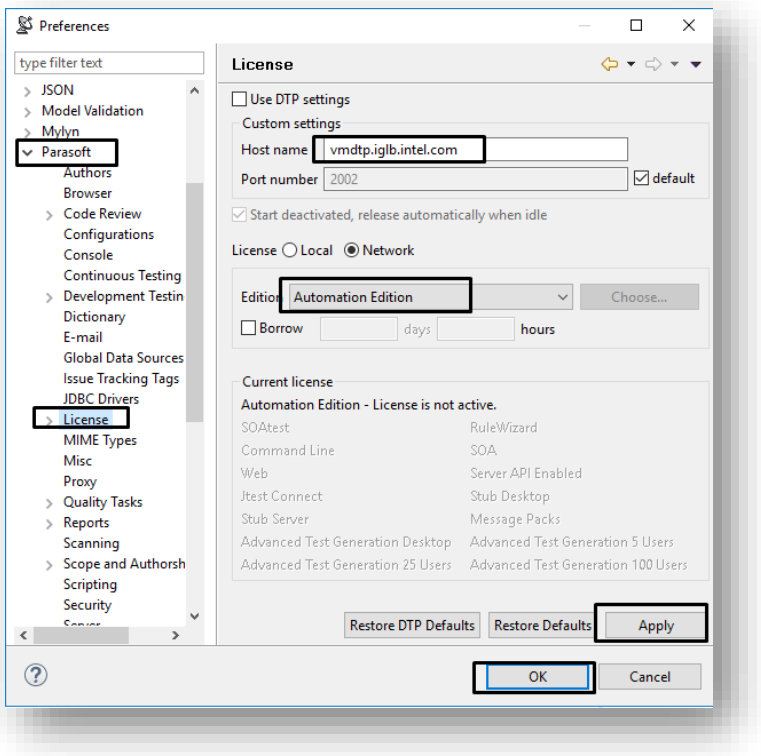




- Go to Window (in the top menu) and click preferences.

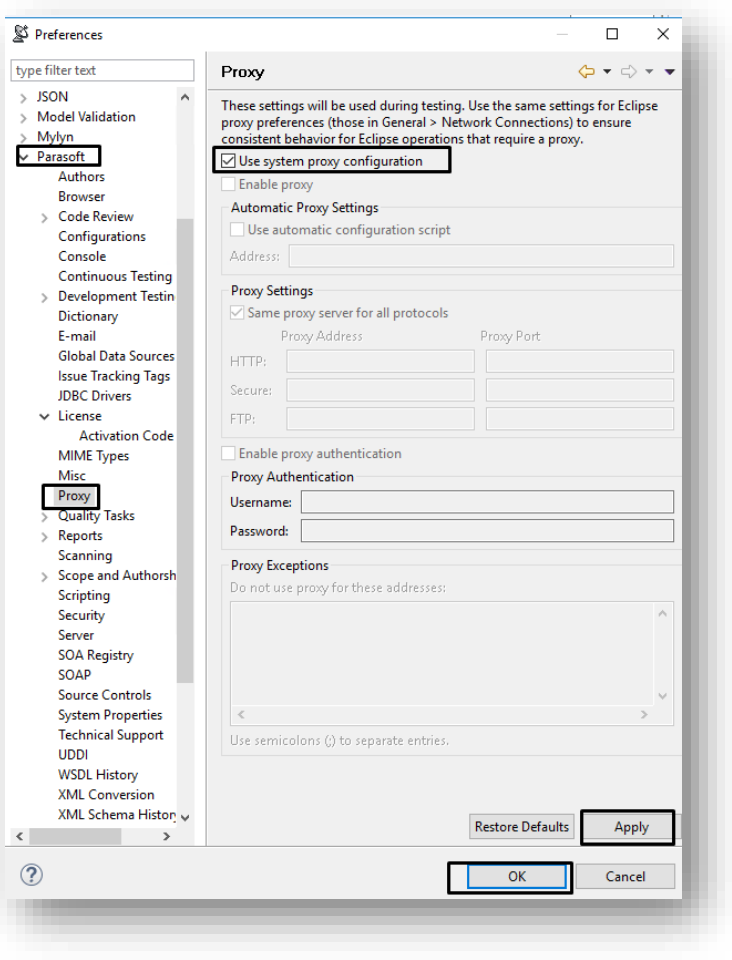


- Navigate and Expand Parasoft → License
    - Enter the hostname: vmdtp.iglb.intel.com
    - Select the “Automation Edition” on the “Edition” Dropdown.
    - Click Apply. Click Ok



- Navigate and Expand Parasoft → Proxy

- Select the option “use system proxy configuration”



6 Set up Gitlab Runner

6.1 Install and register Gitlab Runner

- Create a folder in C drive with name “Gitlab-Runner”
    - C:\GitLab-Runner
- Download the binary amd64 in your local computer. Copy the executable into the Shared folder we created in the first steps. And in the server copy the executable

into the folder you created for gitlab-runner. Rename the binary to gitlab-runner.exe.

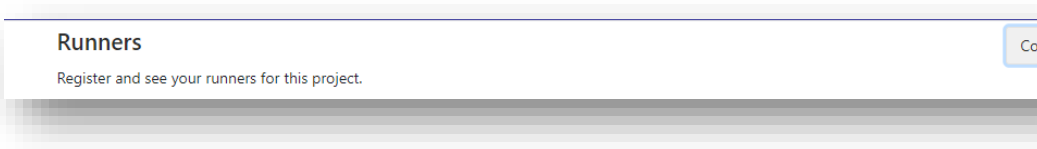
- Press Windows key or click Start button.
- Type PowerShell.
- Right-click Windows PowerShell, and then select Run as administrator.
- In PowerShell
    - Navigate to the folder you created

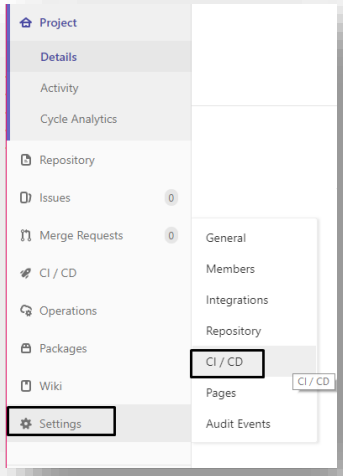
      *Cd *

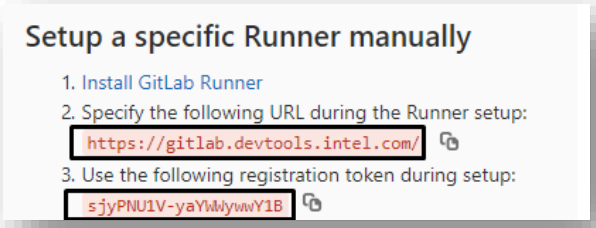
      Cd .\gitlab-runner
    - Register the Runner

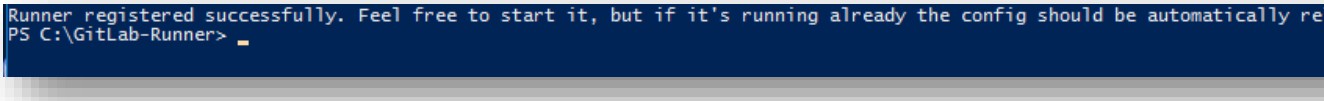
      ./gitlab-runner.exe register

        - Enter the Gitlab-ci coordinator URL: https://gitlab.devtools.intel.com/
        - Enter the Gitlab-ci token. You can find the token in GitLab (settings → runners)
        - Enter Gitlab-ci description: Icost soatest runner (Can be anything you want)
        - Enter Gitlab-ci tags: SOATest,Virtualize
        - Enter the executor: Shell
        - After the success message appears you have the runner registered

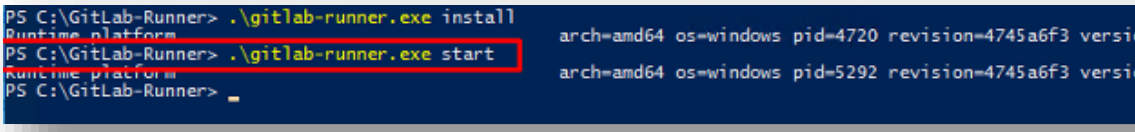



Project
Details
Activity
Cycle Analytics
Repository
Issues
0
Merge Requests
0
CI / CD
Operations
Packages
Wiki
Settings
General
Members
Integrations
Repository
CI / CD
Pages
Audit Events
CI / CD

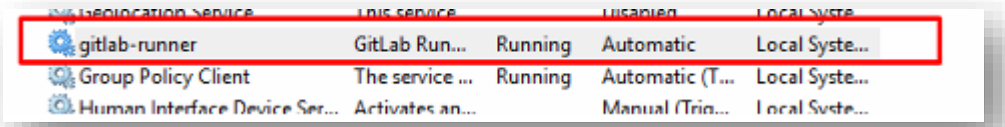

Setup a specific Runner manually
1. Install GitLab Runner
2. Specify the following URL during the Runner setup:
https://gitlab.devtools.intel.com/
3. Use the following registration token during setup:
sjyPNU1V-yaYWWywwY1B


Runner registered successfully. Feel free to start it, but if it's running already the config should be automatically re
PS C:\GitLab-Runner>

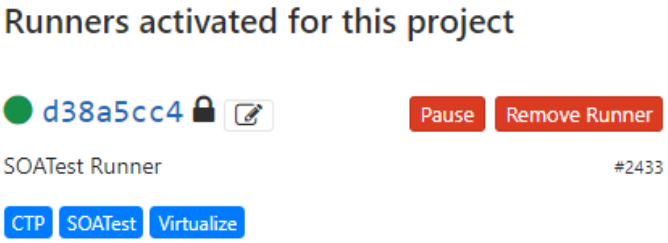
- In PowerShell again
  - Install the Runner: ./gitlab-runner.exe install
  - And Start the service: ./gitlab-runner.exe start



- Open Component Services and verify the gitlab-runner service is running

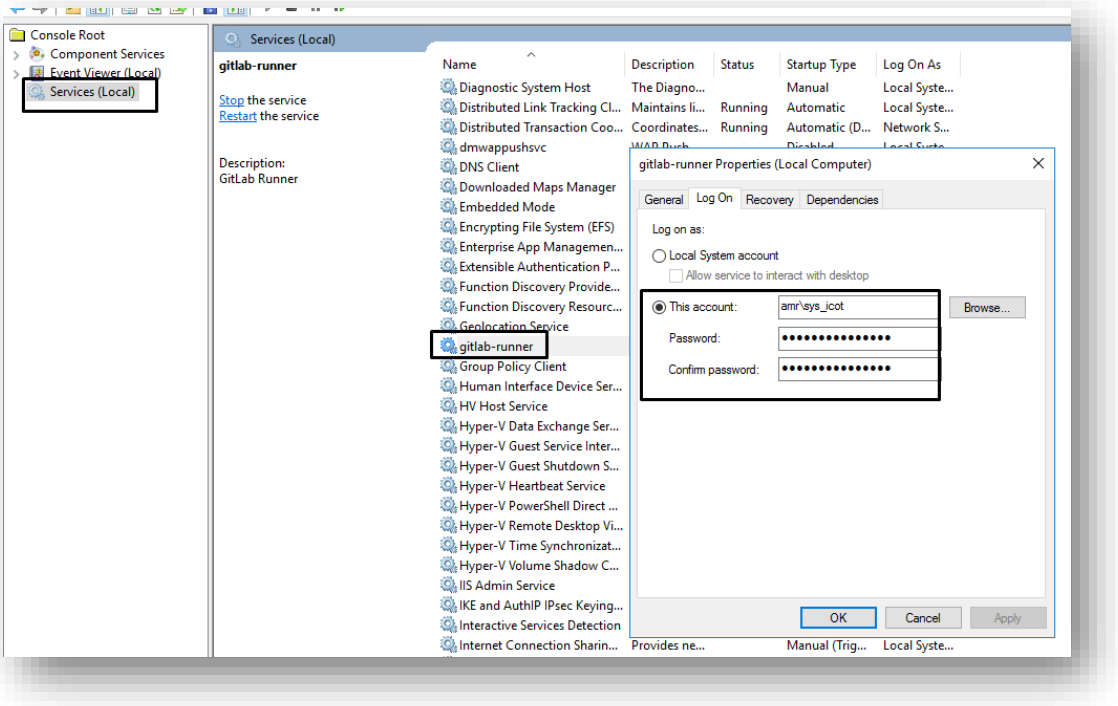


- Make sure the runner was registered correctly in gitlab. Settings → CI/CD → Runners



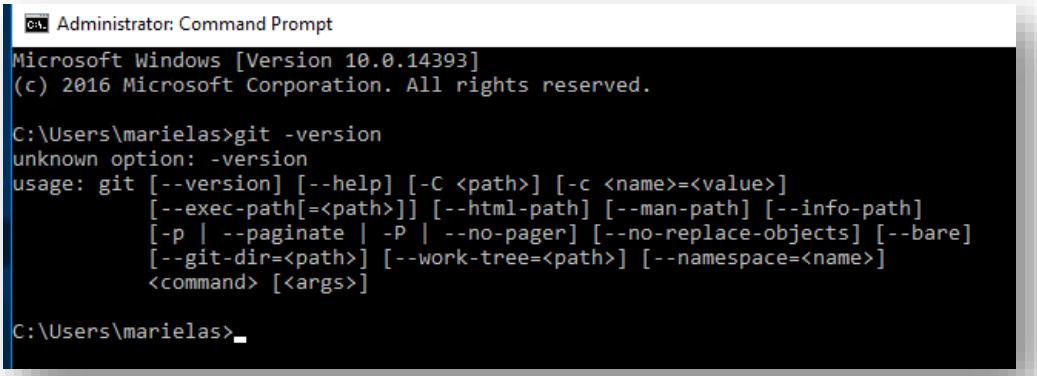
6.2 Configure Gitlab Runner account

- In the server, open component services. Services
- Right click on Gitlab-runner. Properties
- Add the system account and password. Click ok
- Restart the service.



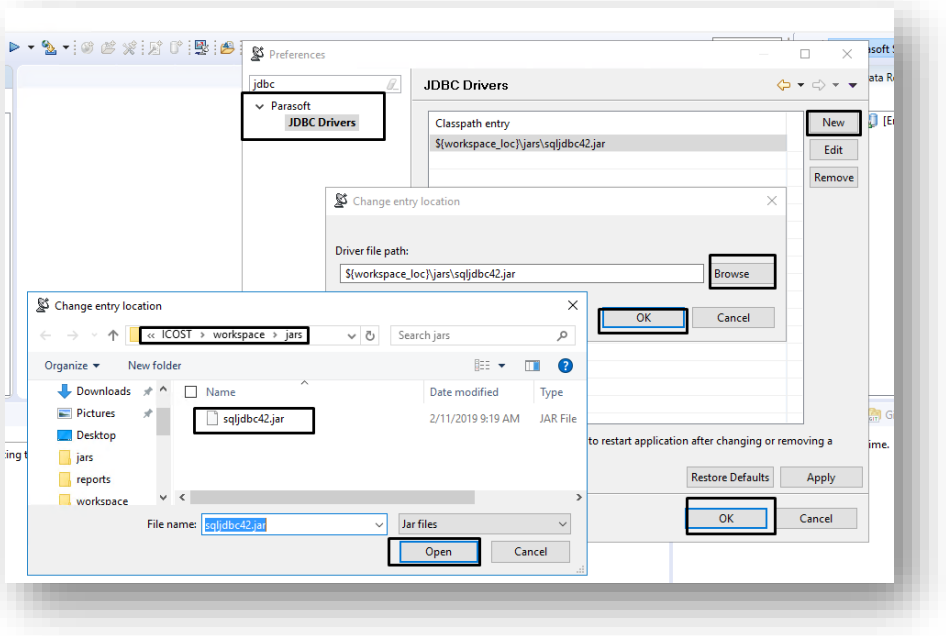
6.3 Install Git in the Server

- Download in your local machine GIT for windows https://git-scm.com/download/win
- Copy the executable into the Shared folder of the server
- In the server. Run the executable
- Install with defaults.
- Make sure the installation was successful.
- Please go to component services and restart the gitlab-runner service to avoid any issue.



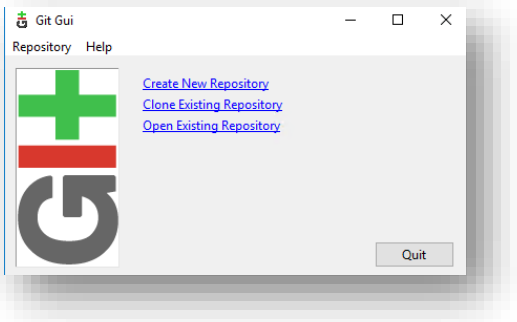
6.4 Add required files to SOATest

- Download in your local machine dll (sqljdbc_auth.dll) and jar (sqljdbc42.jar) files https://wiki.ith.intel.com/display/ITQM/Database+Connections
- Copy the files to the shared folder of the server
- Copy the **dll** file to C:/windows/system32
- And the **jar** file copy to the workspace in a new folder called "jars" C:\ICOST\workspace\jars
- Open SOAtest
- Go to the menu. Windows → Preferences
- Parasoft → JDBC driver
- Click add. Browse the jar file in the workspace → jars folder
- Select the jar file. Click Open. Click ok and finally click ok.
- You can close SOATest

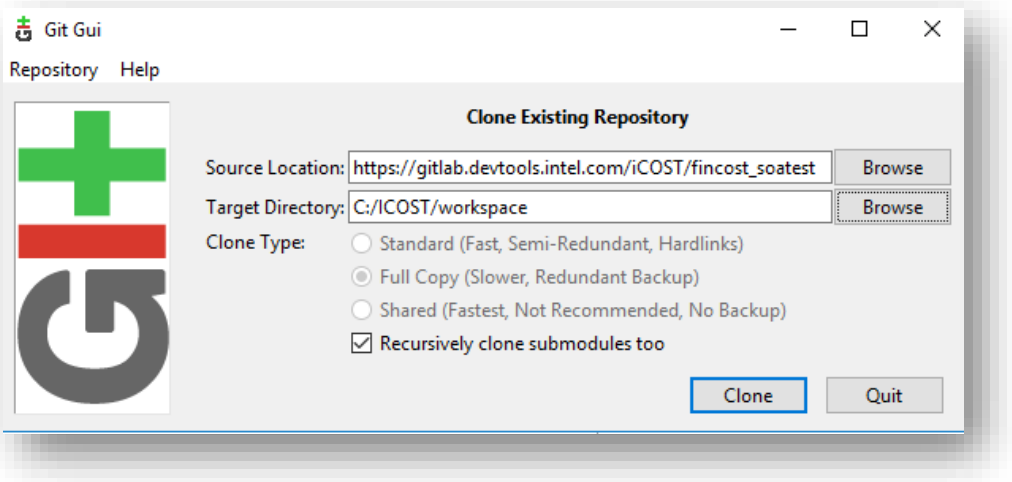


6.5 Clone SOAtest Scripts Repository in the Server

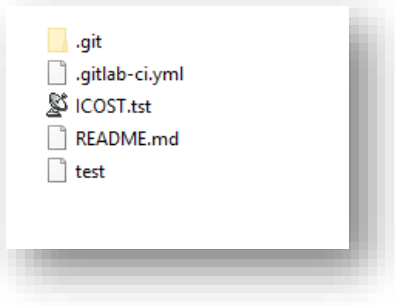
- You can use any git tool for cloning, merging, commit, and others.
- In this case I will use GIT GUI. Click Clone existing repository



- Input source location: https://gitlab.devtools.intel.com/Icost/fincost_soatest
- And target directory will be the workspace we created for SOATest: C:/ICOST/workspace. Click clone

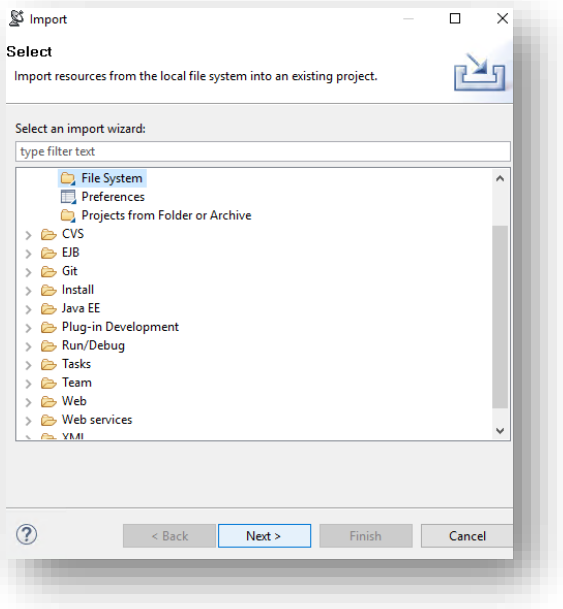


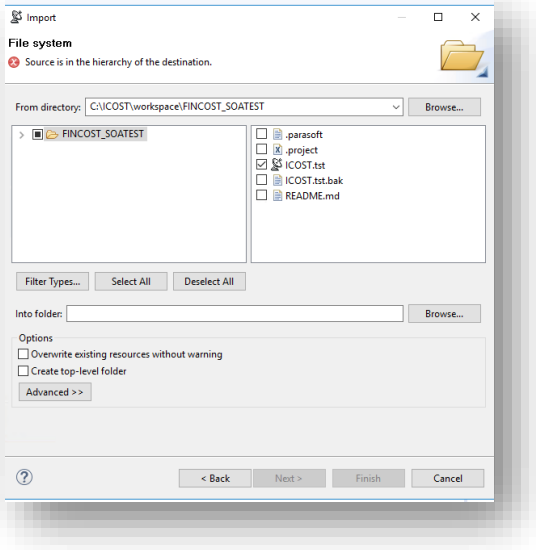
- And target directory will be the workspace we created for SOATest: C:/ICOST/workspace/FINCOST_SOATEST (the folder fincost_soatest should not exists)
- Make sure you get the files from the repository



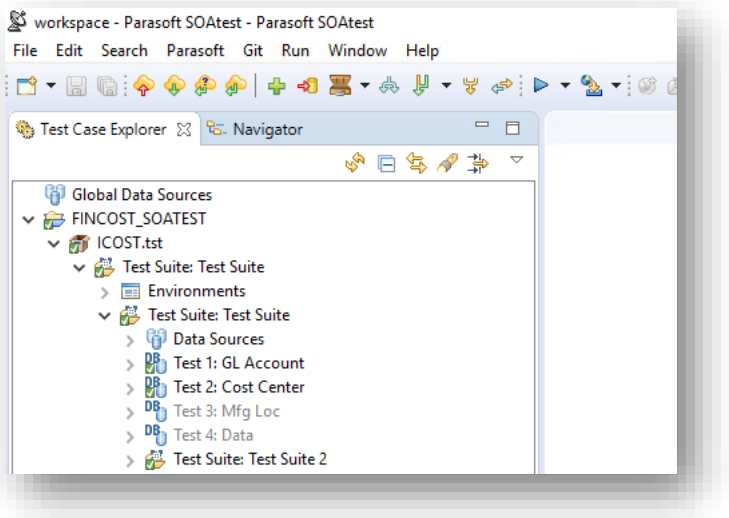
6.6 Import SOAtest Scripts into SOATest

- Open SOATest
- Click File → Import
- Keep File system and click next
- Browse the repository folder. C:\ICOST\workspace\FINCOST_SOATEST
- Click all files with extension .tst. In this case right now there is just one file.
- Click finish





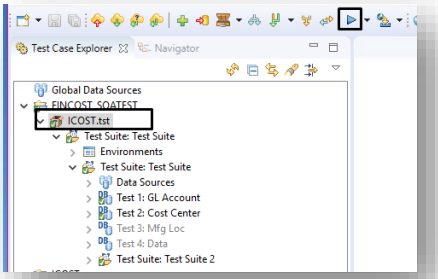
- In SOATest you will see the project already loaded.

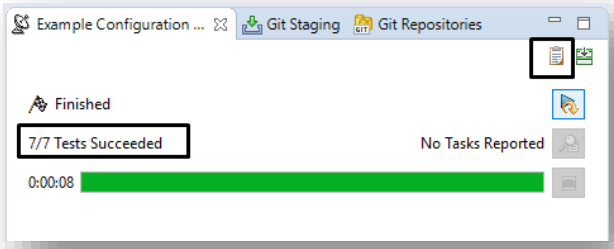


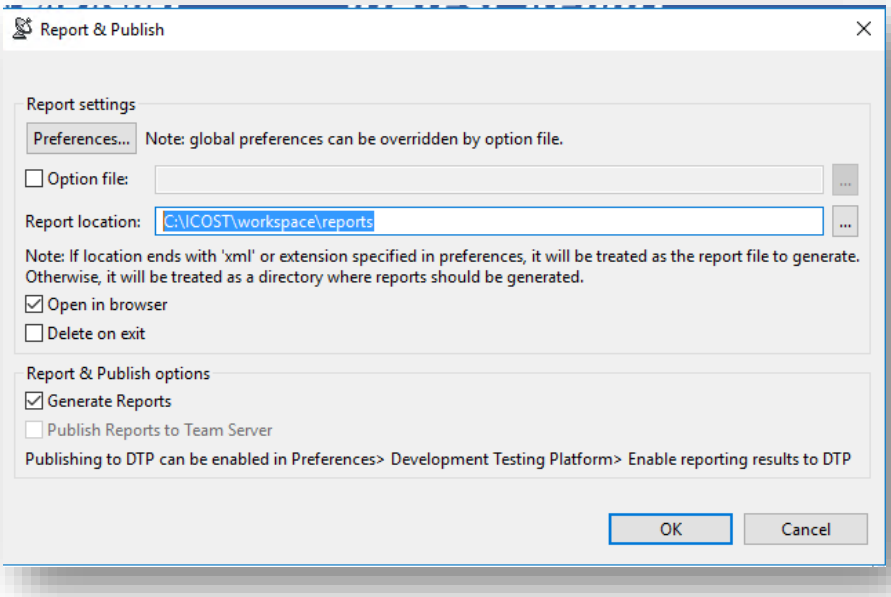
7 Configure and Run CI/CD

7.1 Set up SOATest Reports

- Open SOATest
- Select ICOST.tst or any other test and run it.
- On the botton right section. Click Generate Report
- Report Location: C:\ICOST\workspace\reports (you need to create folder called reports inside the workspace folder). Click Ok

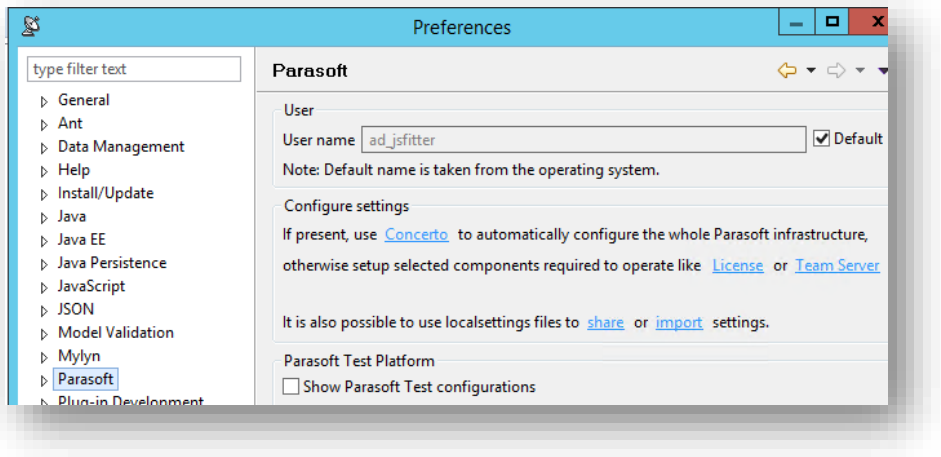

Test Case Explorer
Navigator
Global Data Sources
FINCOST_SOATEST
ICOST.tst
Test Suite: Test Suite
Environments
Test Suite: Test Suite
Data Sources
Test 1: GL Account
Test 2: Cost Center
Test 3: Mfg Loc
Test 4: Data
Test Suite: Test Suite 2


Example Configuration ...
Git Staging
Git Repositories
Finished
7/7 Tests Succeeded
No Tasks Reported
0:00:08

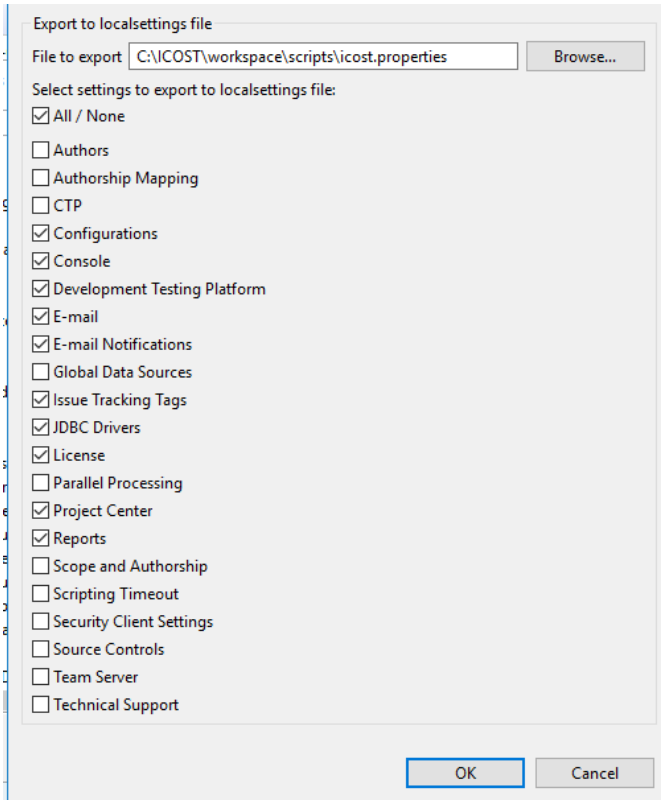

Report & Publish
Report settings
Preferences...
Note: global preferences can be overridden by option file.
Option file:
Report location:
C:\ICOST\workspace\reports
Note: If location ends with 'xml' or extension specified in preferences, it will be treated as the report file to generate. Otherwise, it will be treated as a directory where reports should be generated.
Open in browser
Delete on exit
Report & Publish options
Generate Reports
Publish Reports to Team Server
Publishing to DTP can be enabled in Preferences> Development Testing Platform> Enable reporting results to DTP
OK
Cancel

7.2 Set up .bat SOAtest file

- Launch SOAtest. Goto → Windows → Preferences → Parasoft → Click "share" link to the right.



- Select the location where the properties file will reside. Create a new folder named "Property" under the workspace.
- File name can be anything but must have the extension of ".properties". C:\ICOST\workspace\scripts\Icost.properties
- Below are the most common settings. Click OK



- Create a .bat file in the same folder created above. C:\ICOST\workspace\scripts\Icost.bat
- Copy this code:

C:

cd

cd "Program Files\Parasoft\SOAtest\9.10"

SET arg1=%1

SET arg2=%2

REM SET arg1=C:\ICOST\workspace\reports\report_Icost.html

REM SET arg2=Dev

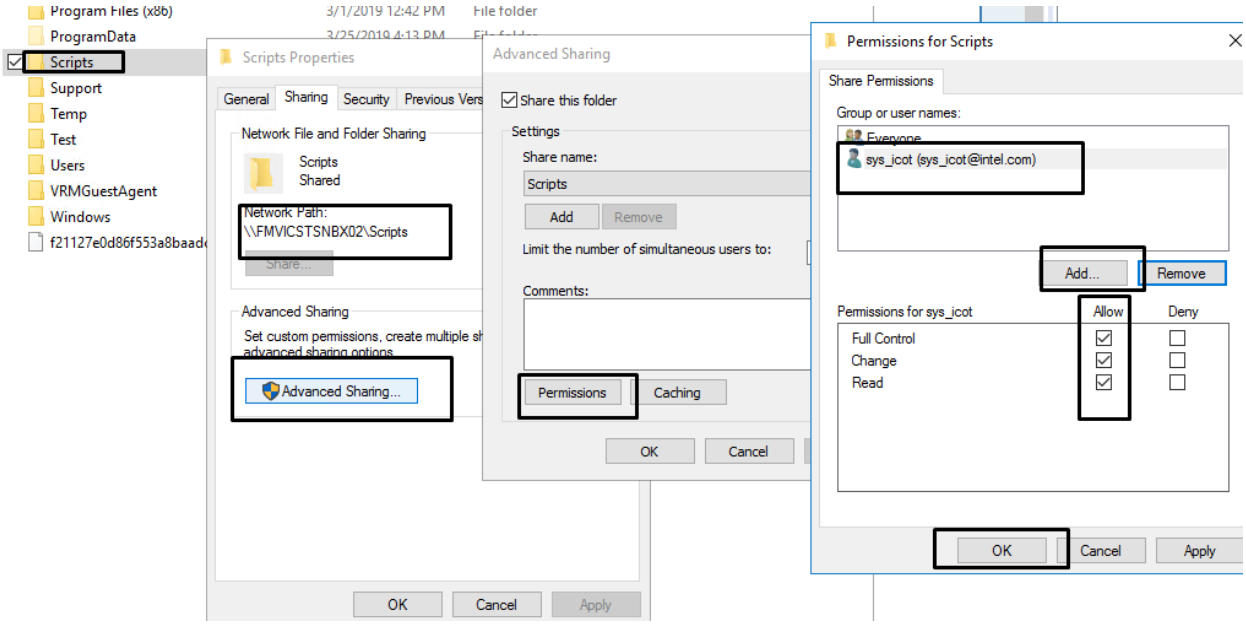
SET localsettings=C:\Scripts\Icost.properties

SET test=/FINCOST_SOATEST/ICOST.tst

SET workspace=C:\ICOST\workspace

soatestcli.exe -data "%workspace%" -resource "%test%" -config "builtin://Demo Configuration" -localsettings "%localsettings%" -report %arg1% -environment %arg2%

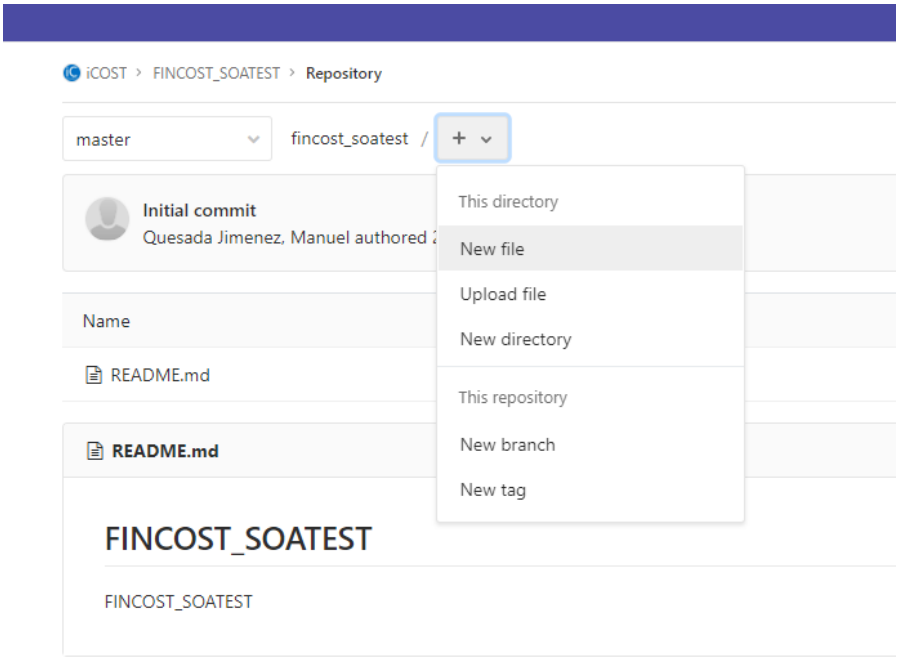
- Share the folder scripts to the system account. \\FMVICSTSNBX02\Scripts



7.3 Gitlab Set up

Yml file

- Open GITLAB. Example: https://gitlab.devtools.intel.com/Icost/fincost_soatest
- Create a new file. Template YML
- Name the file .gitlab-ci.yml



- Name of the file .gitlab-ci.yml
- Copy the following code:

stages:

- test

- reports

#This below command is run usind the soatestcli executable from the vmssoavrt001 server....

SOAP/REST Command Line:

stage: test

tags:

- SOATest

- CTP

- Virtualize

script: \\FMVICSTSNBX02\Scripts\Icost.bat C:\ICOST\workspace\reports\report_Icost.html Dev

 # stage: reports

 # script: echo "Report is located at C:\GitLab-Runner\Reports\SoapRestCommandline.html on vmssoavrt001 server"

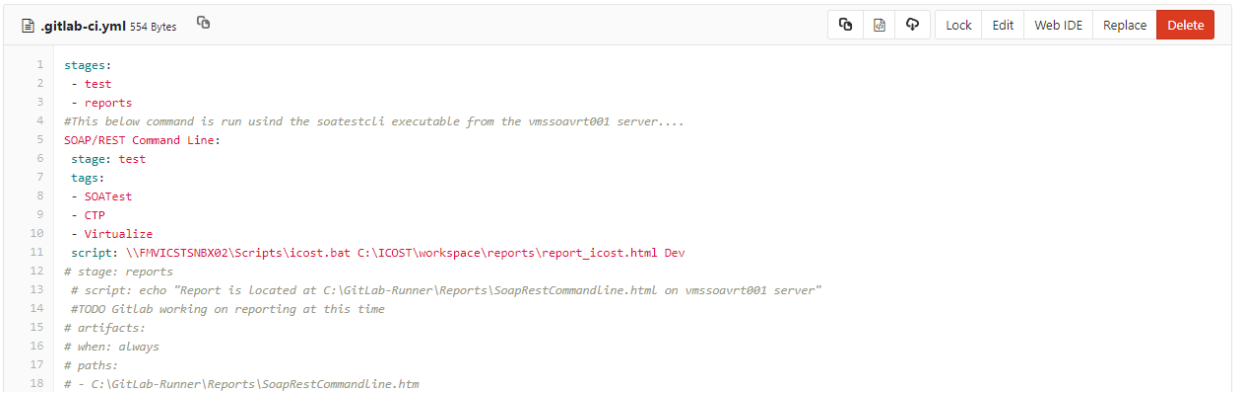
#TODO Gitlab working on reporting at this time

 # artifacts:

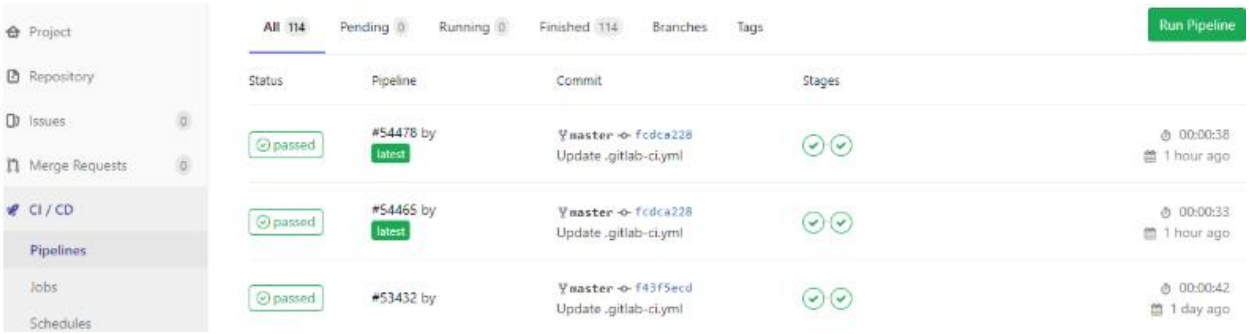
 # when: always

 # paths:

 # - C:\GitLab-Runner\Reports\SoapRestCommandline.htm



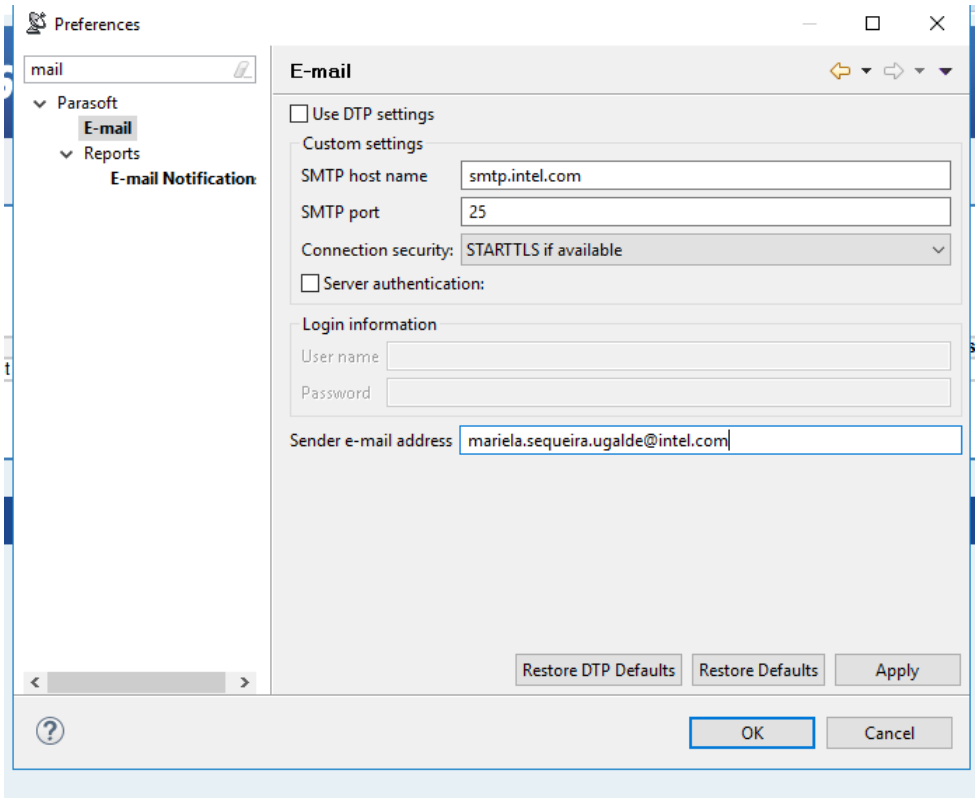
- Test by doing a commit to the project. You can go to CI/CD → Pipelines



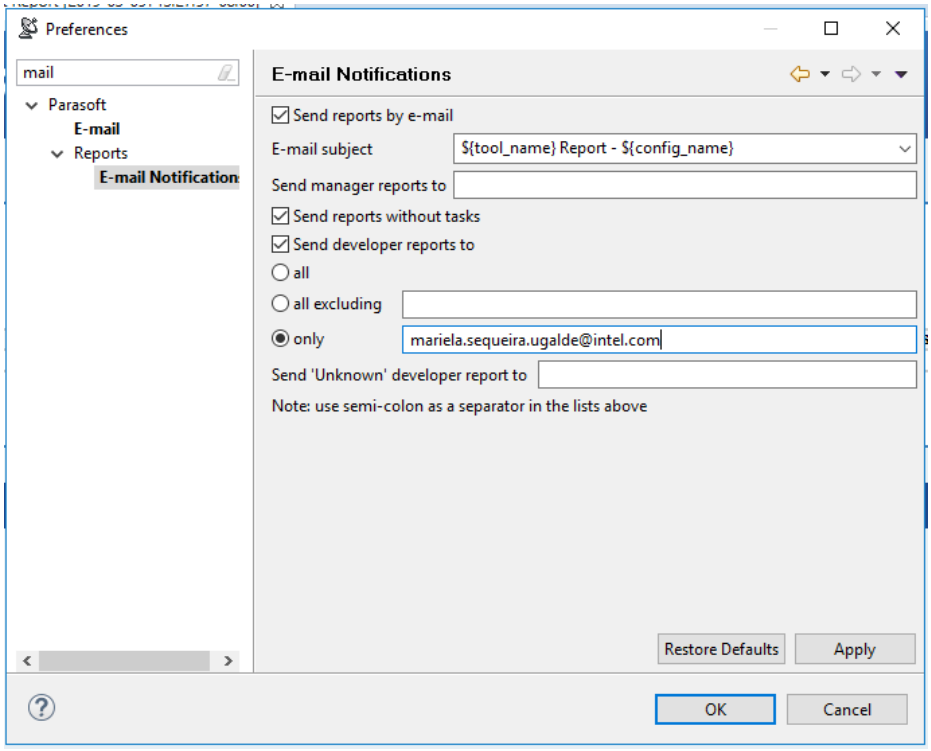
7.4 Set up email configuration in SOAtest

- Launch SOAtest → Window → Preferences → Parasoft → Email
  - SMTP Host Name = smtp.intel.com
  - SMTP port = 25

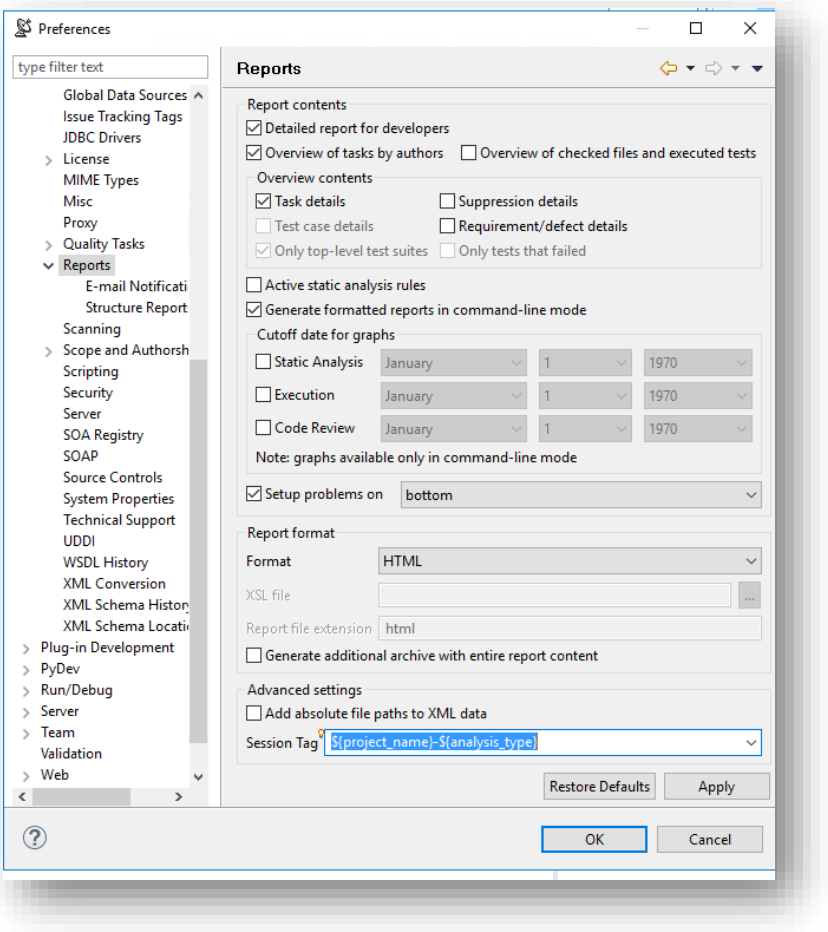
- Sender e-mail Address = noreply@intel.com or another valid email address



- Window → Preferences → Parasoft → Email notification
- Email-subject select ${tool_name} Report - ${config_name}
- Send reports to: any email or group account



- Window → Preferences → Parasoft → Reports
- Make sure the session tag is ${project_name}-${analysis_type}



Anexo 26 - Manual de usuario de SOATest

Este anexo se encuentra en idioma inglés y es dirigido a los desarrolladores de ICOST con el fin de brindarles una guía y un paso a paso de cómo instalar la herramienta y cómo utilizar el marco de trabajo realizado.

SQL STORE PROCEDURE TESTING USING SOATEST

Juan Diego González Arias, Mariela Sequeira Ugalde

Contents

8 Version History

Ver. #	Date	Updated by	Comments
Draft	03/04/2020	Mariela Sequeira Ugalde Juan Diego González Arias	First Draft Created

9 Scope

This document covers the best practices for testing SQL Stored Procedures in the Icost project using SOAtest tool.

10 Target Audience

The target audience of this document is Developers and SA's who are able to work with SQL queries. Technical notes are provided for Dev and Technical users who have the skills to enhance the solution framework to accommodate any new or changing needs.

10.1 Unit Testing Workflow

Unit testing requires a blank DB with only static lookup data loaded into it. This process needs to be repeatable.

The flow for Unit tests for SQL is

1. Before each Unit test, inject necessary data
2. Run each Unit test (store procedure or SQL)
3. After each Unit test, delete any injected data

10.2 Introducing SOATest

Automate complete end-to-end testing for business and security-critical transactions. Parasoft SOAtest is widely recognized as the leading enterprise-grade solution for API testing and API integrity. Thoroughly test composite applications with

robust support for REST and web services, plus an industry-leading 120+ protocols/message types.

Key Features

- ✓ Recognized for “ease for use”, intuitive interface
- ✓ Advanced intelligent automated test generation
- ✓ Extensive protocol and technology support
- ✓ End-to-end testing across multiple endpoints (services, ESBs, databases, JMS, EDI, mainframes, web UI, ERPs…)
- ✓ Designed to support continuous testing
- ✓ Integrates with GitHub. Recommended to use GitHub as a repository and for source control.

SOAtest is the recommended tool for API and web service testing at Intel IT. SOAtest is a good tool to have as part of a SAAS product test strategy.

11 Setting up your Environment for SQL Unit Testing

11.1 Prerequisites

- You will need to have access to Gitlab project so that you can clone the project and check in changes.
  - Request access in AGS https://ags.intel.com and search for “GitLab Users”
  - One approval has completed, you can logon at https://gitlab.devtools.intel.com/ or http://goto.intel.com/gitlab
  - Contact your administrator Icost.dba@intel.com and request access to SOATest Icost project “FINCOST_SOATEST”

- You will need some knowledge of how to use SOATest and Gitlab.
  - Request access to SOATest
  - https://ags.intel.com/identityiq/home.jsf search for and Request to the entitlement "SOATest License Read Only".

- Once approved (it takes about a day for AGS to propagate. This will give access to the fileshare and license)
- Once approved you will be able to have access to the share to download the training material and SOATest Install. (Note: you can start the training and will not need the software until you are ready for the exercises)
  - **Download software and install:** \\10.2.67.165\SOATest\SOALicense
  - **Training:** \\10.2.67.165\SOATest\SOATraining

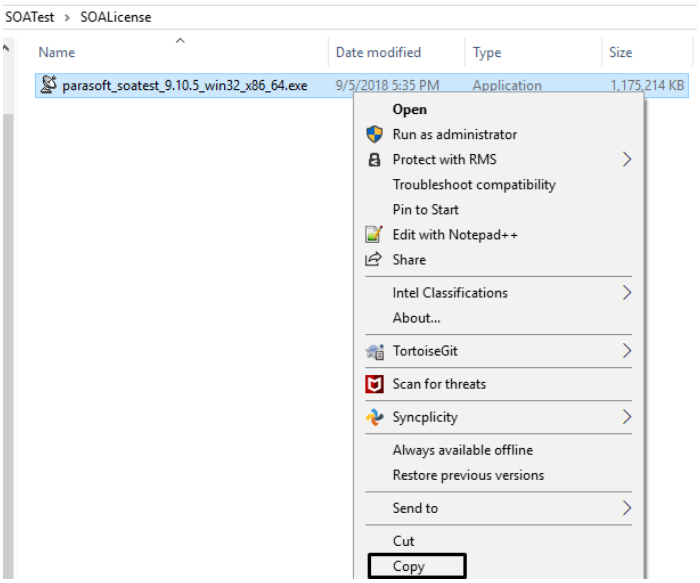
- Go to SABA to take the training https://intel.sabacloud.com/Saba/Web_spf/NA2PRD0003/common/searchresults/soatest/ALL
  - IT Test Automation_SOATest Prerequisites
  - IT Test Automation_SOATest Automating API Tests

- You will need Tortoise or any other GIT tool.

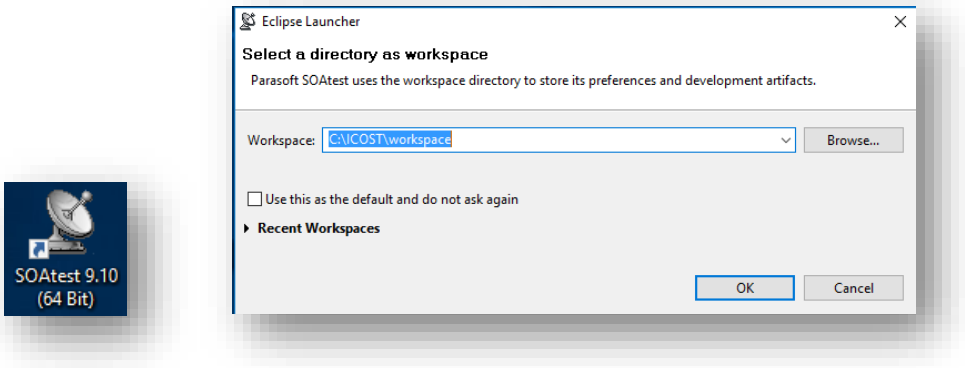
11.2 Set-Up Steps

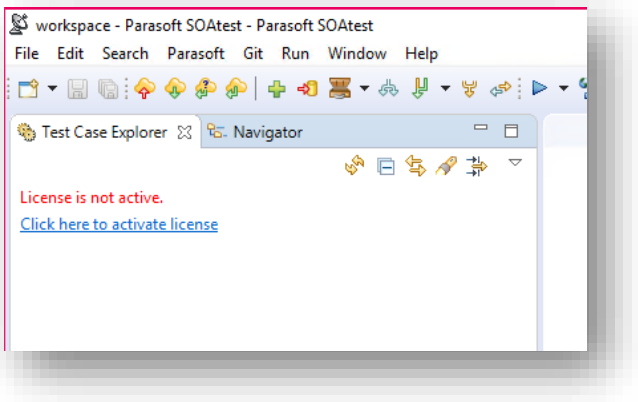
11.2.1 Install and Set up SOATest

- Create a folder (in desktop) with any name. This folder will help us to copy the executables of the programs that we need to install.
- Open a new file explorer and navigate to
  - AMR → \\10.2.67.165\SOATest\SOALicense
  - GAR → \\10.109.66.4\SOATest\SOALicense
  - Or ask mariela.sequeira.ugalde@intel.com to provide you the SOATest installer

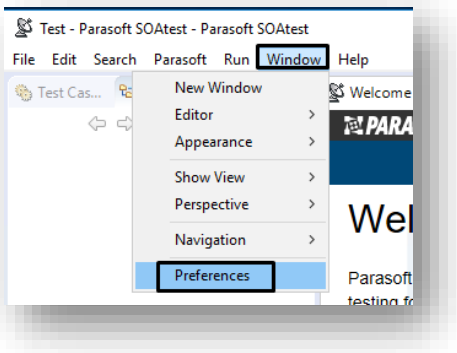


- Copy the .exe file and paste the file in the folder created in the first step
- Run the executable as an administrator. (Right Click on the File)
- Accept license terms and follow the install wizard instructions to install SOAtest.
- Make sure checkbox is checked to create shortcut on your desktop to launch. Use all the defaults.
- Launch SOATest
    - Double Click on the SOATest Icon on your desktop
    - Input the workspace: C:\ICOST\workspace
    - Click Ok
    - If you receive the message “License is not active”. Click the link to activate the license.



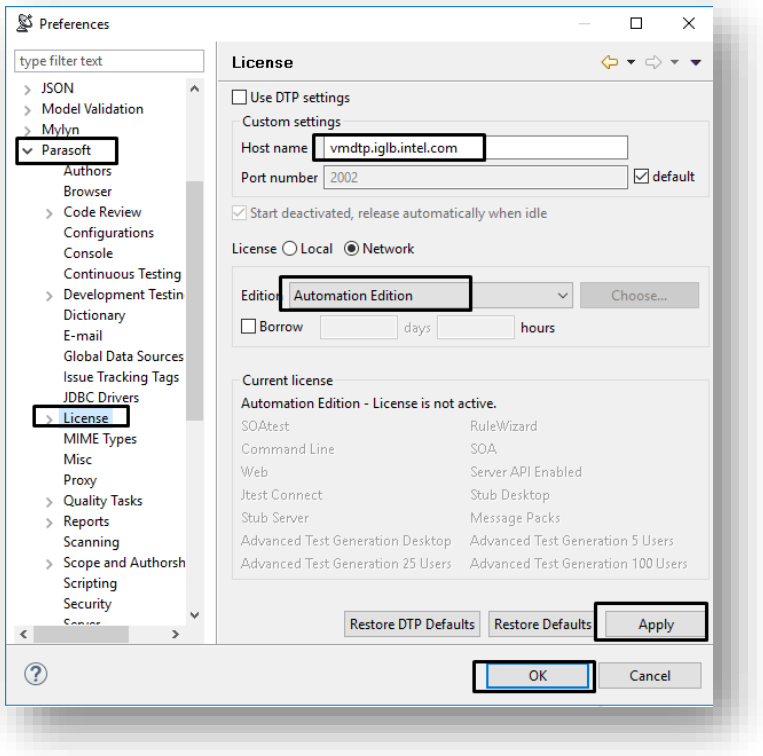


- Go to Window (in the top menu) and click preferences.

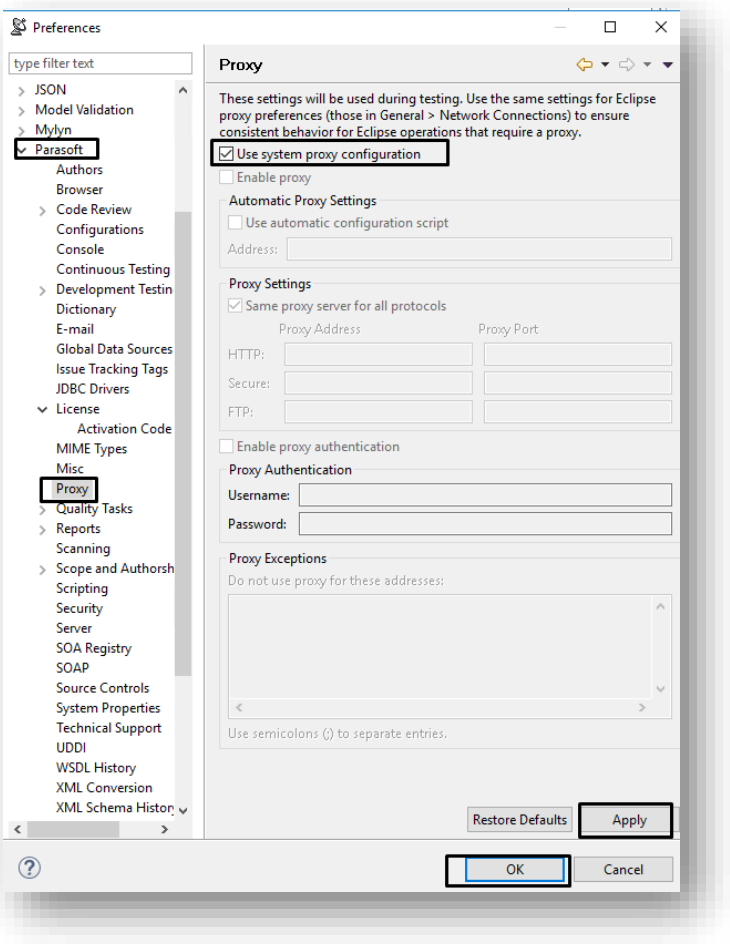


- Navigate and Expand Parasoft → License
    - Enter the hostname: vmdtp.iglb.intel.com
    - Select the “Automation Edition” on the “Edition” Dropdown.

- Click Apply. Click Ok

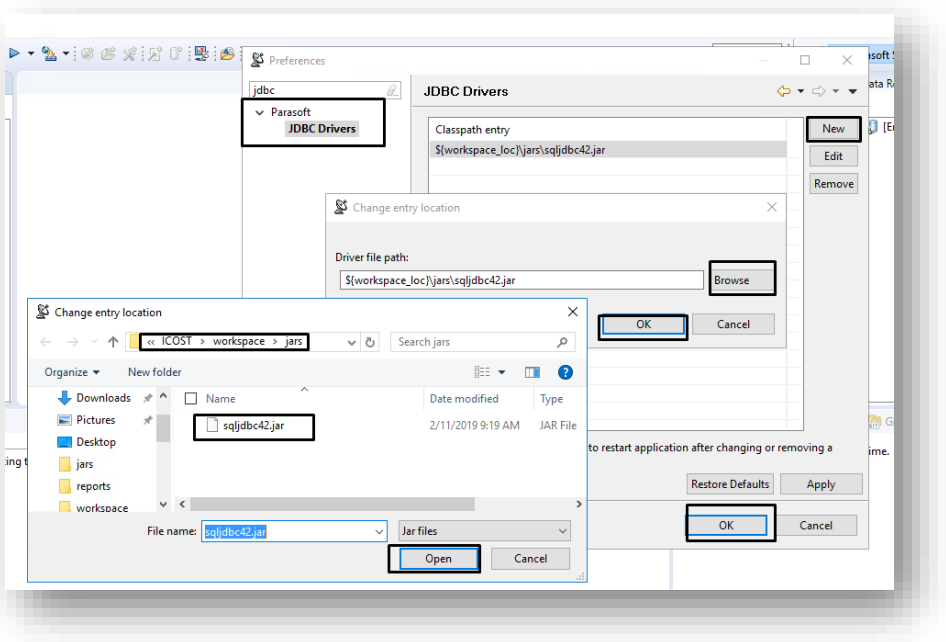


- Navigate and Expand Parasoft → Proxy
    - Select the option “use system proxy configuration”



- Download in your local machine dll (sqljdbc_auth.dll) and jar (sqljdbc42.jar) files https://wiki.ith.intel.com/display/ITQM/Database+Connections
- Copy the **dll** file to C:/windows/system32
- And the **jar** file copy to the workspace in a new folder called "jars" C:\ICOST\workspace\jars
- Open SOAtest
- Go to the menu. Windows → Preferences
- Parasoft → JDBC driver
- Click add. Browse the jar file in the workspace → jars folder
- Select the jar file. Click Open. Click ok and finally click ok.

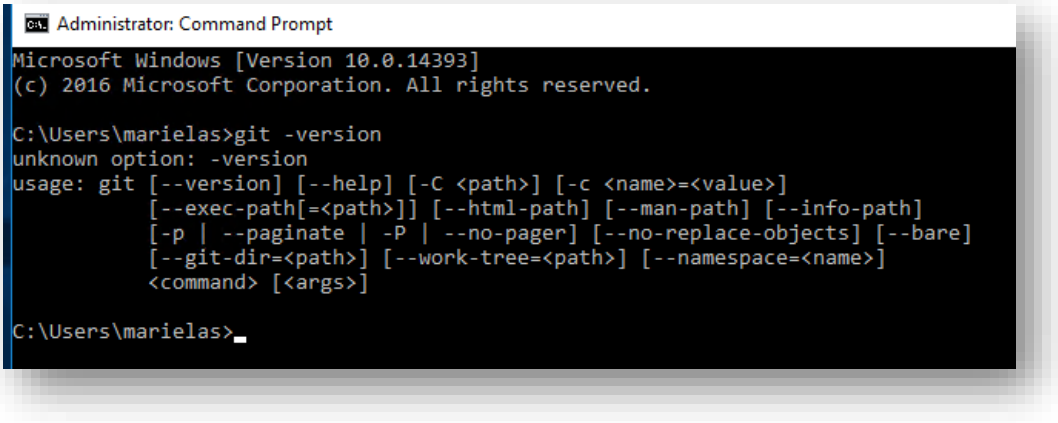
- You can close SOATest



11.2.2 Get the project from Source Control (Gitlab)

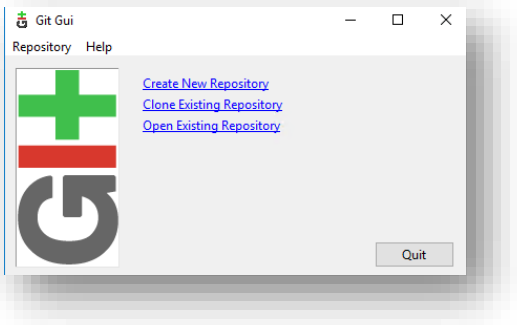
11.2.2.1 Install GIT

- Download GIT for windows https://git-scm.com/download/win
- Run the executable
- Install with defaults.
- Make sure the installation was successful.

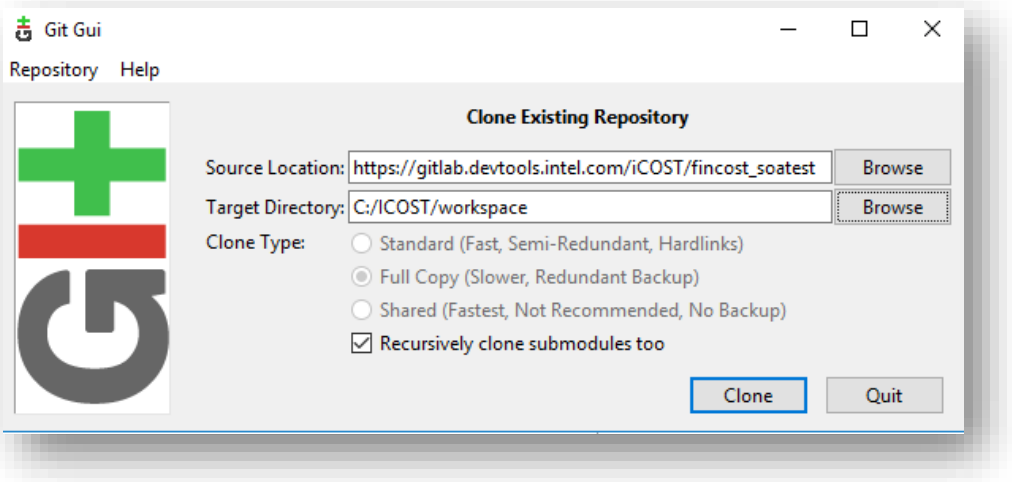


11.2.2.2 Clone GITLAB Project

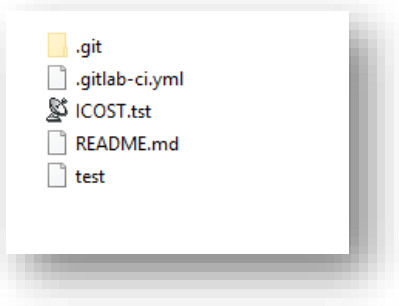
- You can use any git tool for cloning, merging, commit, and others.
- In this case I will use GIT GUI. Click Clone existing repository



- Input source location: https://gitlab.devtools.intel.com/Icost/fincost_soatest
- And target directory will be the workspace we created for SOATest: C:/ICOST/workspace. Click clone

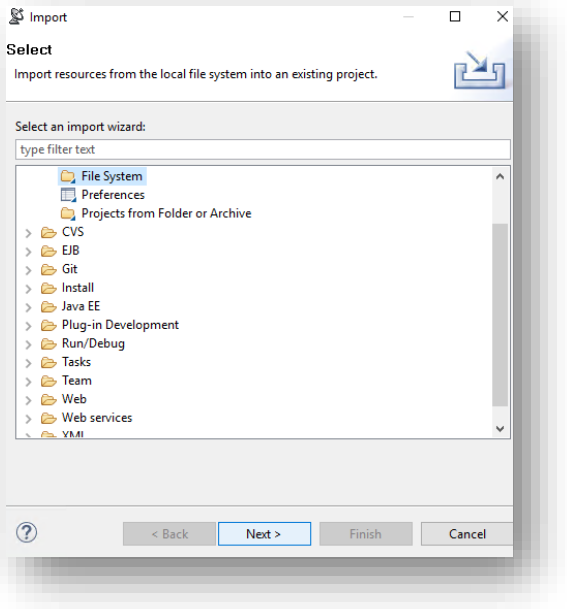


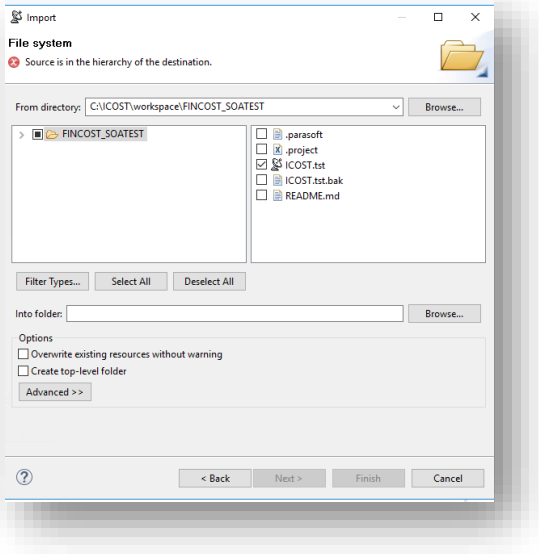
- And target directory will be the workspace we created for SOATest: C:/ICOST/workspace/FINCOST_SOATEST (the folder fincost_soatest should not exists)
- Make sure you get the files from the repository



- Open SOATest
- Click File → Import
- Keep File system and click next
- Browse the repository folder. C:\ICOST\workspace\FINCOST_SOATEST
- Click all files with extension .tst. In this case right now there is just one file.

- Click finish



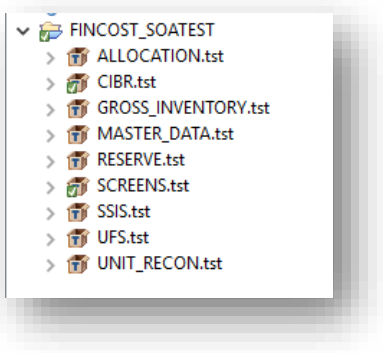


- In SOATest you will see the project already loaded.

11.2.3 Explore the Solution

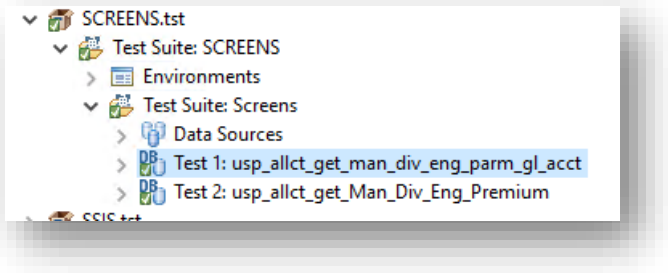
FINCOST_SOATEST is the project in general, contains all tst files.

Tst files are containers of tests, in this case we will use tst files as ICOST functional areas.

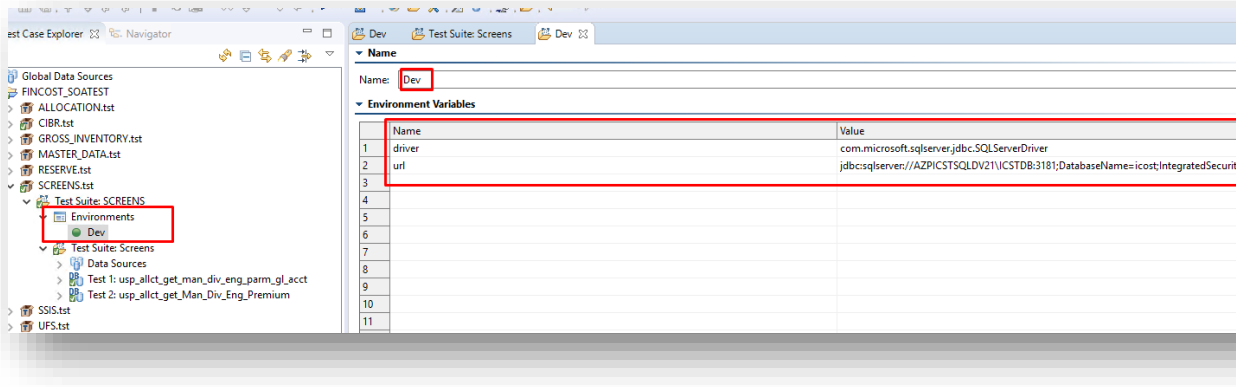


Each tst or test files can contain multiple test suites which contains the tests itself.

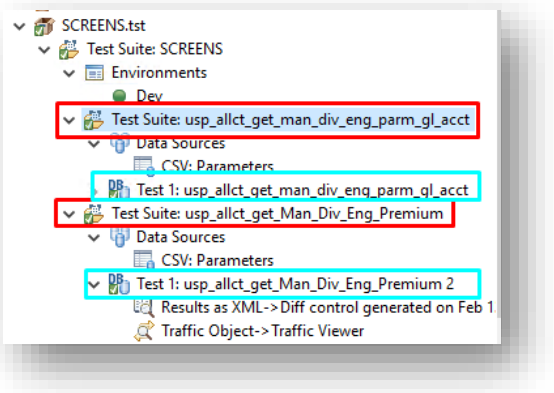
Test Suites is used to add the test of your store procedure, it contains environments, data sources, DB tests, set up and tear down tests.



Environments are used to add any DB environment needed in this image you can see DEV and in the right panel you can see the database information, you can add PS, PROD and others.



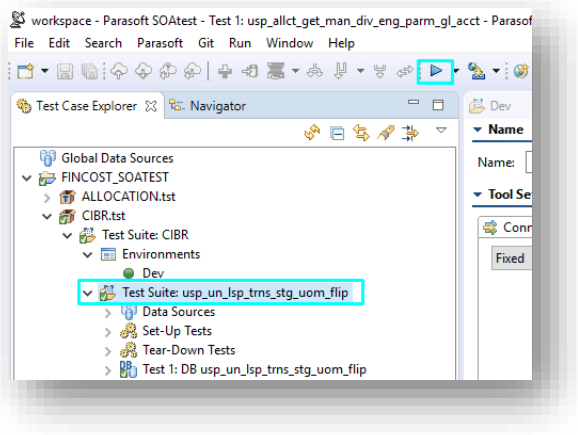
Test suites contains multiple tests, in this scenario tests suites are created one for each by store procedure as you can see in the image. And then inside the test suite you can have multiple tests depending of what you need to test.

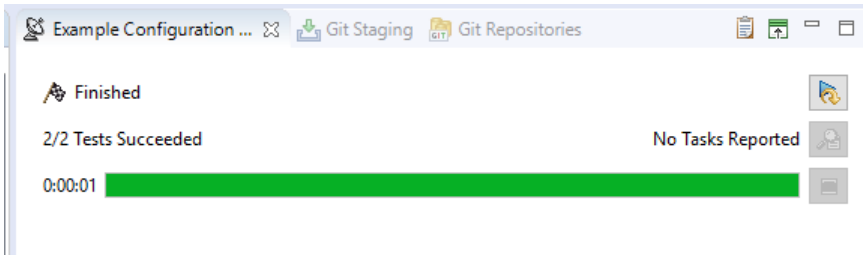


12 Running and Writing Tests

12.1 Running Tests

Select the Test suite, or test or tst file and hit the button run (play button) and check the results on bottom panel.



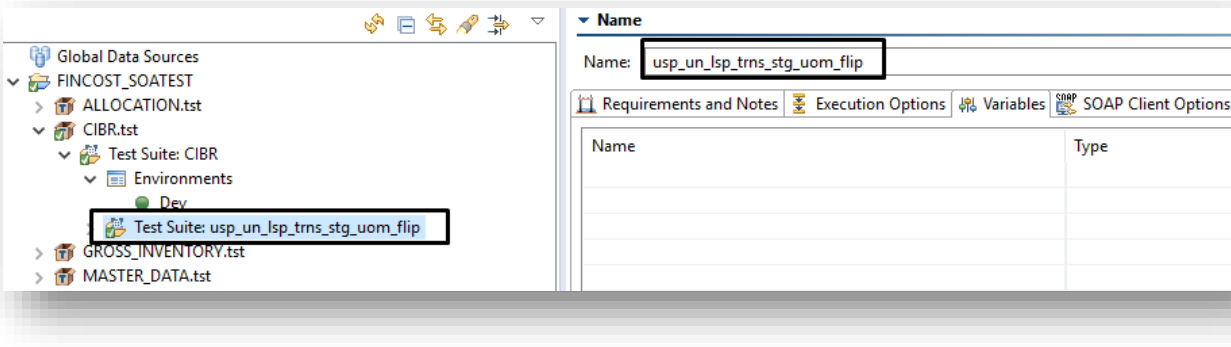


12.2 Writing Tests

12.2.1 To Write a Test for a New Stored Procedure or One with No Tests Yet

12.2.1.1 Nomenclature.

Create a new test suit under the functional area test suit/ test. The name of the test suite should be the name of the store procedure. Then you can add as many tests that you want with a meaningful name of what the test is doing.



12.2.1.2 Create end to end test

First thing you need to realize is that you need to create couple of SQL queries to validate the store procedure. The pattern will be:

1. The tool will have to insert one/several records in some table
2. Execute the SP to pick up the data we just inserted
3. Run a query to validate that the data we process is correct

Example of a test SP: staging.dbo.usp_un_lsp_trns_stg_uom_flip

Insert a dummy record for a year other than production ➔ staging.dbo.un_cibr_trns_stg

- In this sample we will use a record from February 2016

Prerequisites:

Ensure SP dependent tables and rule engine entries are loaded

Test execution:

Execute the SP submitted with parameters to consider only period 2016-02

Note: some SP will not work this way, so the source table will have to be cleared before testing

Test validation:

Check the dummy record with a query to see if it worked or not

Clean up:

Delete the dummy record from the system

SQL queries example:

Row to insert:

```
INSERT INTO dbo.un_cibr_trns_stg
    ([rec_cre_dtm]
    ,[rec_cre_uid]
    ,[rec_upd_dtm]
    ,[rec_upd_uid]
    ,[fscl_yr_nbr]
    ,[fscl_mo_nbr]
    ,[lsp_id]
    ,[strt_trns_gmt]
    ,[rvrs_trns_gmt]
    ,[end_trns_gmt]
    ,[cre_dtm]
    ,[last_upd_dt_tm]
    ,[co_cd]
    ,[lgl_ent_cd]
    ,[fr_to_lsp_id]
    ,[lot_nbr]
    ,[lot_ownr_cd]
    ,[strt_trns_lcl_dtm]
    ,[end_trns_lcl_dtm]
    ,[site_cd]
    ,[to_site_cd]
    ,[origin_site_cd]
    ,[itm_id]
    ,[cost_itm_id]
    ,[compnt_cpu_itm_id]
    ,[prd_nm]
    ,[cost_prd_nm]
    ,[prev_mfg_opr_nbr]
    ,[mfg_opr_nbr]
    ,[src_verb_cd]
    ,[trns_sts_cd]
    ,[uom_cd]
    ,[scnd_uom_cd]
    ,[src_site_cd]
    ,[site_type_cd]
    ,[to_site_type_cd]
```

,[trns_qty]

,[scnd_qty]

,[unrstc_eoh_qty]

,[cum_shp_rcv_qty]

,[bloc_eoh_qty]

,[consig_eoh_qty]

,[yld_loss_qty]

,[stack_qty]

,[bonus_qty]

,[excrsn_loss_qty]

,[inv_stok_loss_qty]

,[src_sys_nm]

,[rt_nm]

,[intel_po_nbr]

,[wrhse_intrns_flag]

,[wrhse_rcv_trns_gmt]

,[wrhse_rcv_trns_lcl_dtm]

,[cust_nbr]

,[mtrl_doc_nbr]

,[mtrl_doc_line_nbr]

,[dlvr_note_nbr]

,[atrb_val]

,[itm_type_cd]

,[mfg_loc_cd]

,[to_mfg_loc_cd]

,[cost_mfg_stg_cd]

,[mfg_prcss_nm]

,[mfg_dot_prcss_nm]

,[rwrk_ind]

,[ignr_excpt_ind]

,[miss_trns_id]

,[rvlt_ind]

,[rvrs_trns_ind]

,[prft_centr_chg_ind]

,[engnr_to_prod_ind]

,[prod_to_engnr_ind]

,[true_miss_gene_ind]

,[exec_trns_ind]

,[cam_modul_ins_ind]

,[new_lot_ind]

,[stg_chg_ind]

,[shp_ind]

,[rcv_ind]

,[ur_exec_ind]

,[inv_pnt_ind]

,[rec_id]

```
    ,[supl_opr_nbr]
    ,[adj_po_nbr]
    ,[adj_po_line_nbr]
    ,[shp_mdia_cd]
    ,[prd_type_cd]
    ,[plnt_type_cd]
    ,[prcss_cd]
    ,[inv_type_cd]
    ,[src_verb_grp_id]
    ,[adj_src_verb_cd]
    ,[job_id]
    ,[job_param_dtm]
    ,[ignr_rsn_id]
    ,[prcss_cmt_txt])
VALUES
    (cast('02/05/2016 04:13:00' as smalldatetime) --[rec_cre_dtm]
    ,'AMR\autosys' --[rec_cre_uid]
    ,getdate() --[rec_upd_dtm]
    ,'flip logic test' --[rec_upd_uid]
    ,2016 --[fscl_yr_nbr]
    ,2 --[fscl_mo_nbr]
    ,'723879793' --<lsp_id, varchar(12),>
    ,cast('02/03/2016 16:42:05' as smalldatetime) --<strt_trns_gmt, datetime,>
    ,NULL --<rvrs_trns_gmt, datetime,>
    ,cast('02/05/2016 00:08:22' as smalldatetime) --<end_trns_gmt, datetime,>
    ,cast('02/03/2016 12:06:35' as smalldatetime) --<cre_dtm, datetime,>
    ,cast('02/04/2016 20:06:28' as smalldatetime) --<last_upd_dt_tm, datetime,>
    ,'100' --<co_cd, varchar(10),>
    ,'100' --<lgl_ent_cd, varchar(10),>
    ,NULL --<fr_to_lsp_id, varchar(12),>
    ,'NACAX3070' --<lot_nbr, varchar(15),>
    ,'P' --<lot_ownr_cd, char(1),>
    ,cast('02/04/2016 01:42:05' as smalldatetime) --<strt_trns_lcl_dtm, datetime,>
    ,cast('02/05/2016 09:08:22' as smalldatetime) --<end_trns_lcl_dtm, datetime,>
    ,'BYC' --<site_cd, varchar(10),>
    ,NULL --<to_site_cd, varchar(10),>
    ,NULL --<origin_site_cd, varchar(10),>
    ,'2000-170-292' --<itm_id, varchar(18),>
    ,'2000-170-292' --<cost_itm_id, varchar(18),>
    ,NULL --<compnt_cpu_itm_id, varchar(18),>
    ,'2000-170-292' --<prd_nm, varchar(25),>
    ,'2000-170-292' --<cost_prd_nm, varchar(25),>
    ,'2000' --<prev_mfg_opr_nbr, varchar(8),>
    ,'7000' --<mfg_opr_nbr, varchar(8),>
    ,'MVOU' --<src_verb_cd, varchar(8),>
    ,NULL --<trns_sts_cd, char(2),>
```

```
,'U' --<uom_cd, char(1),>
,'W' --<scnd_uom_cd, char(1),>
,'BYC' --<src_site_cd, varchar(10),>
,'SUB' --<site_type_cd, char(3),>
,NULL --<to_site_type_cd, char(3),>
,9024 --<trns_qty, int,>
,12 --<scnd_qty, int,>
,9024 --<unrstc_eoh_qty, int,>
,0 --<cum_shp_rcv_qty, int,>
,0 --<bloc_eoh_qty, int,>
,0 --<consig_eoh_qty, int,>
,0 --<yld_loss_qty, int,>
,0 --<stack_qty, int,>
,0 --<bonus_qty, int,>
,0 --<excrsn_loss_qty, int,>
,0 --<inv_stok_loss_qty, int,>
,'ADT' --<src_sys_nm, varchar(30),>
,NULL --<rt_nm, varchar(10),>
,'0000000000000NA' --<intel_po_nbr, varchar(20),>
,NULL --<wrhse_intrns_flag, char(2),>
,NULL --<wrhse_rcv_trns_gmt, datetime,>
,NULL --<wrhse_rcv_trns_lcl_dtm, datetime,>
,NULL --<cust_nbr, varchar(11),>
,NULL --<mtrl_doc_nbr, varchar(10),>
,NULL --<mtrl_doc_line_nbr, int,>
,NULL --<dlvr_note_nbr, varchar(18),>
,NULL --<atrb_val, varchar(12),>
,'RAPP' --<itm_type_cd, char(4),>
,'BYC' --<mfg_loc_cd, varchar(10),>
,NULL --<to_mfg_loc_cd, varchar(10),>
,NULL --<cost_mfg_stg_cd, varchar(20),>
,NULL --<mfg_prcss_nm, varchar(10),>
,NULL --<mfg_dot_prcss_nm, varchar(50),>
,0 --<rwrk_ind, bit,>
,0 --<ignr_excpt_ind, bit,>
,0 --<miss_trns_id, int,>
,0 --<rvlt_ind, bit,>
,0 --<rvrs_trns_ind, bit,>
,0 --<prft_centr_chg_ind, bit,>
,0 --<engnr_to_prod_ind, bit,>
,0 --<prod_to_engnr_ind, bit,>
,0 --<true_miss_gene_ind, bit,>
,0 --<exec_trns_ind, bit,>
,0 --<cam_modul_ins_ind, bit,>
,0 --<new_lot_ind, bit,>
,0 --<stg_chg_ind, bit,>
```

```
,0 --<shp_ind, bit,>
,0 --<rcv_ind, bit,>
,0 --<ur_exec_ind, bit,>
,0 --<inv_pnt_ind, bit,>
,NULL --<rec_id, int,>
,NULL --<supl_opr_nbr, varchar(8),>
,NULL --<adj_po_nbr, varchar(10),>
,NULL --<adj_po_line_nbr, char(5),>
,NULL --<shp_mdia_cd, char(4),>
,NULL --<prd_type_cd, varchar(10),>
,NULL --<plnt_type_cd, varchar(10),>
,NULL --<prcss_cd, varchar(14),>
,NULL --<inv_type_cd, char(1),>
,NULL --<src_verb_grp_id, int,>
,NULL --<adj_src_verb_cd, varchar(100),>
,30021 --<job_id, int,>
,cast('02/04/2016 14:32:46' as smalldatetime) --<job_param_dtm, datetime,>
,0 --<ignr_rsn_id, int,>
,'Initial Load,UOM RESET,SCND_UOM RESET,UOM FLIP,UOM FLIP,' --<prcss_cmt_txt, varchar(5000),>
    );
```

Execute SP:

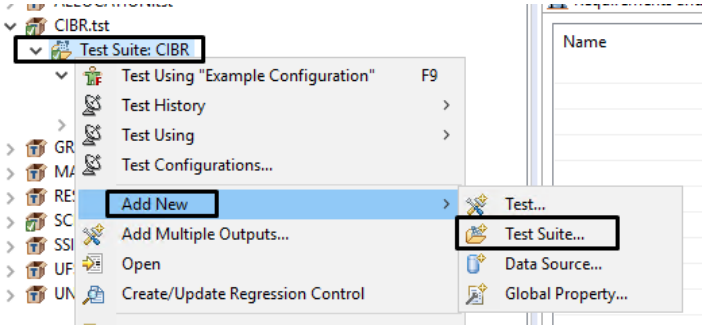
```
EXEC dbo.usp_un_lsp_trns_stg_uom_flip 0, 201602, Null
```

Query staging.dbo.un_cibr_trns_stg and validate that the record that was inserted has the following fields set as this:

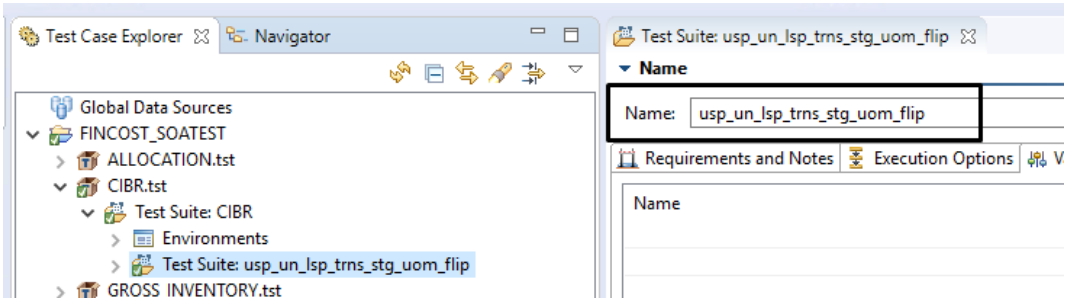
```
select top 1 * from dbo.un_cibr_trns_stg where
uom_cd = 'W' and
scnd_uom_cd = 'U' and
trns_qty=12 and
scnd_qty=9024 and
unrstc_eoh_qty=12
```

so in SOATest in order to automate above example you will need to follow this steps:

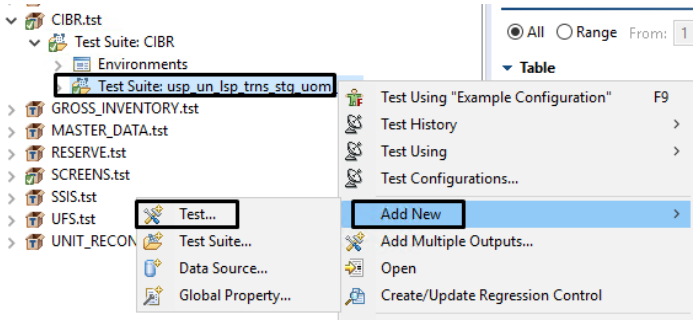
1. Do right click in the Test Suite (functional area)
2. Click add test suite

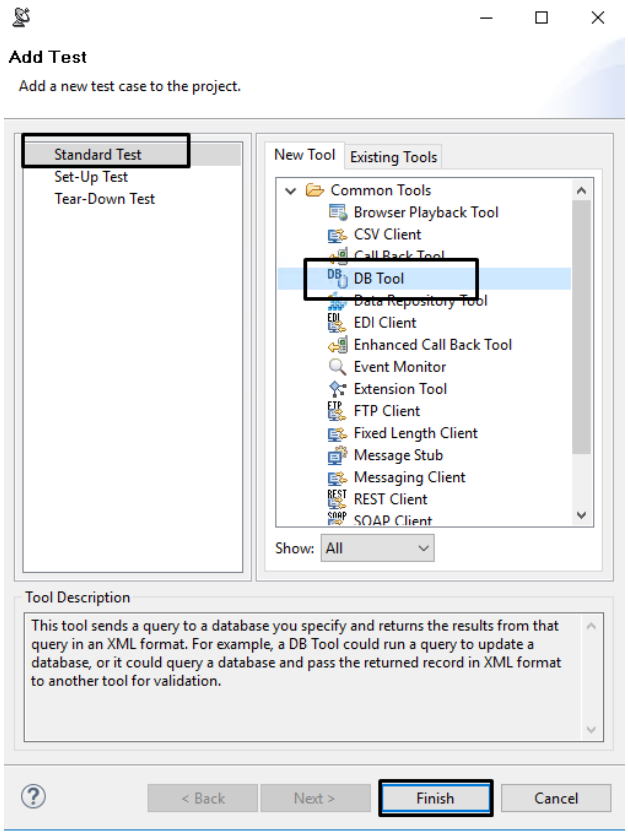


3. Rename with Store Procedure Name as mentioned in the nomenclature

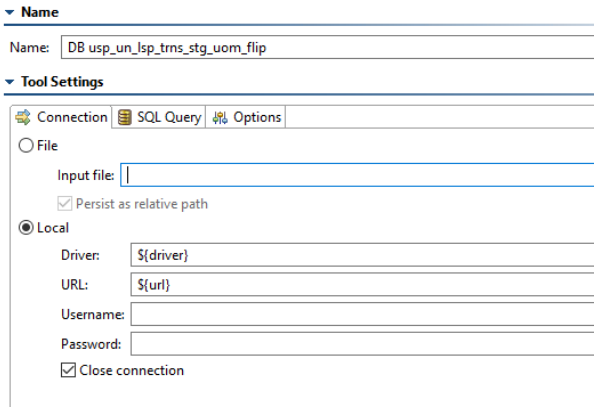


4. Do right click on the test suite created above and click add Test

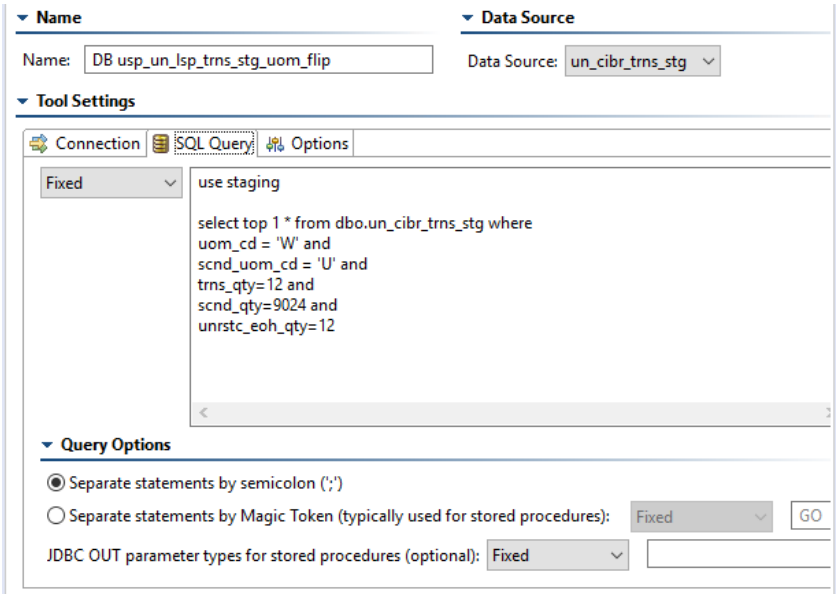




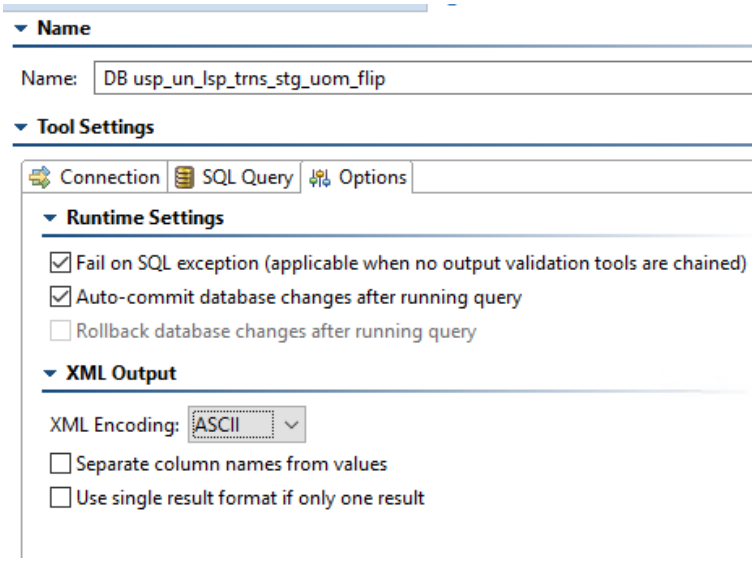
5. Select Standard Test → DB Tool and click Finish
6. Rename the test. Here you can use a meaningful name.
    a. Add driver and URL variable used in the Environments



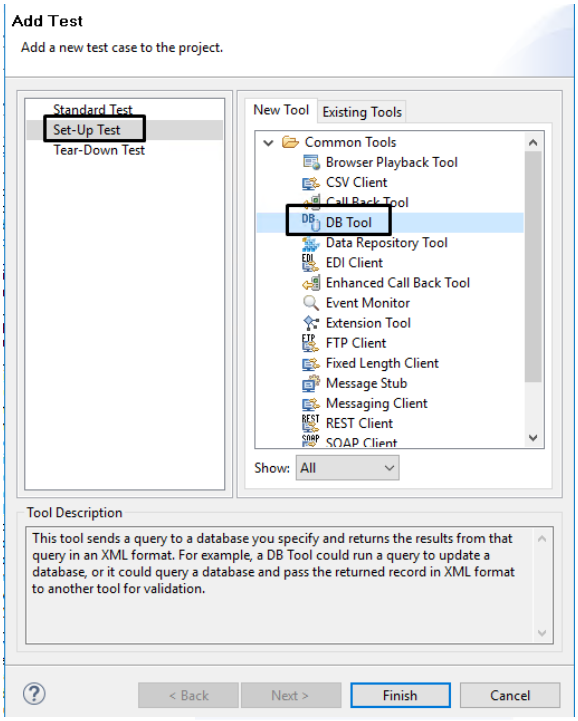
7. Select SQL Query tab. Add the SQL query to validate the store procedure execution

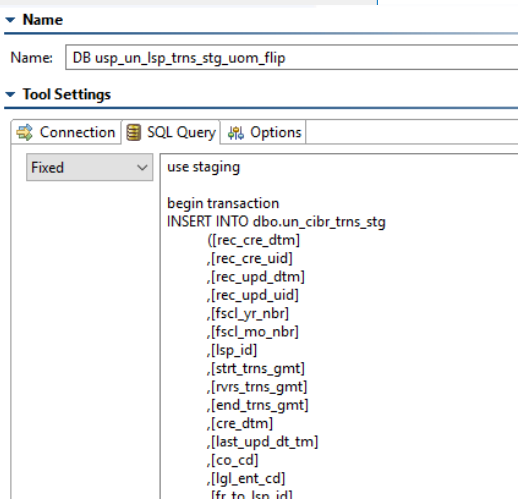


8. Make sure the options looks like the below image.

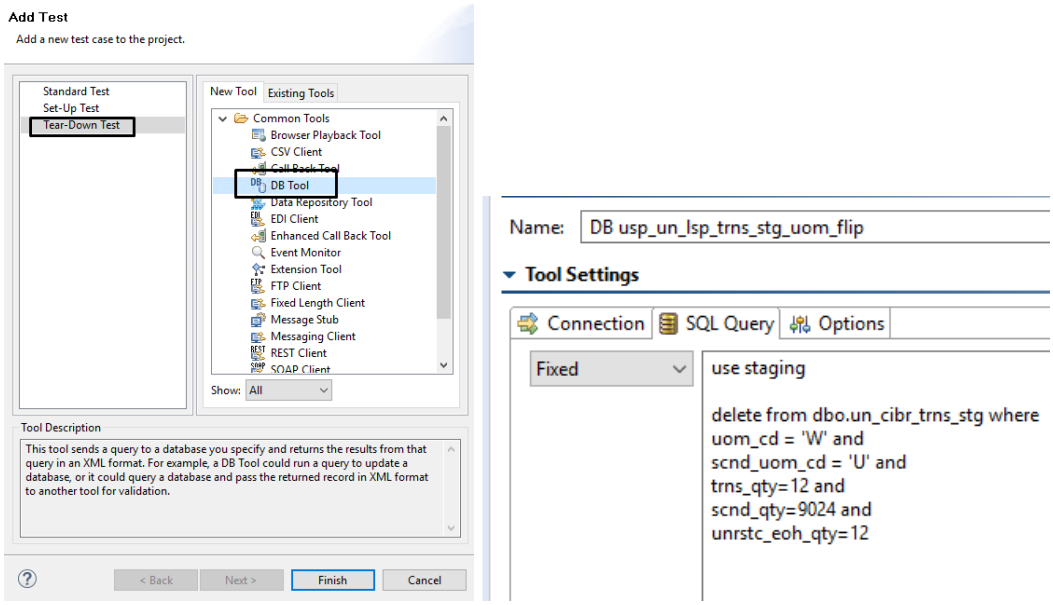


9. Set-up test
    a. Right click test suite, add Test
    b. Select Set-up Test and DB Tool
    c. Click Finish. Rename the test
    d. Add the Insert Query and the Query to execute the store procedure.

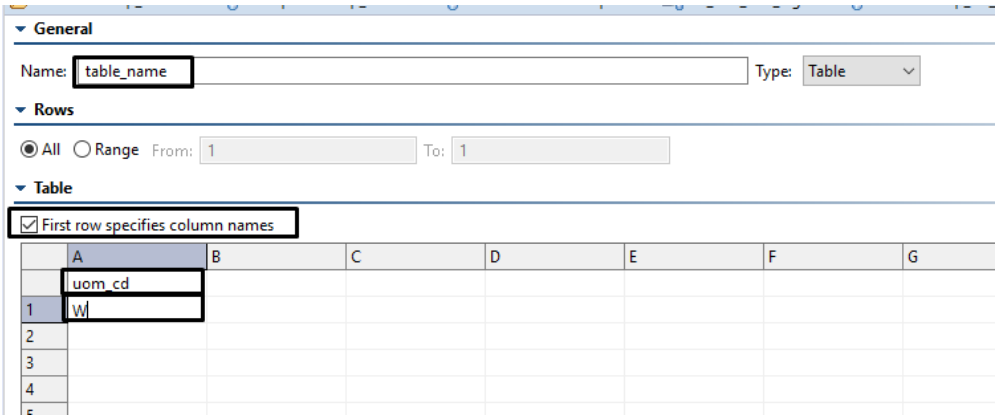




10. Tear down test
    a. Right click test suite, add Test
    b. Select Tear-down Test and DB Tool
    c. Click Finish. Rename the test
    d. Add the delete Query.



11. Add Data Source (Table, CSV, Excel file)
    a. Right click test suite, add Data Source
    b. Select **Table**. Click Finish.
    c. Click checkbox “First row specifies column names”
    d. Add column name and value to compare with

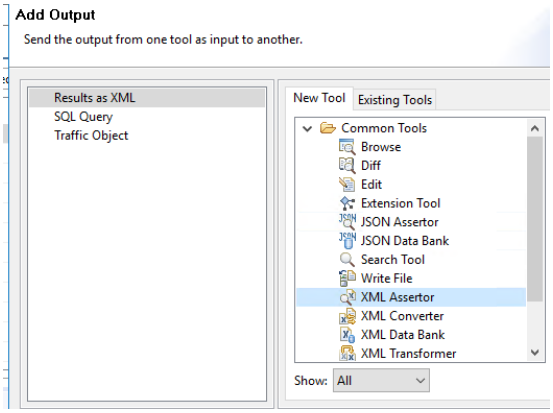


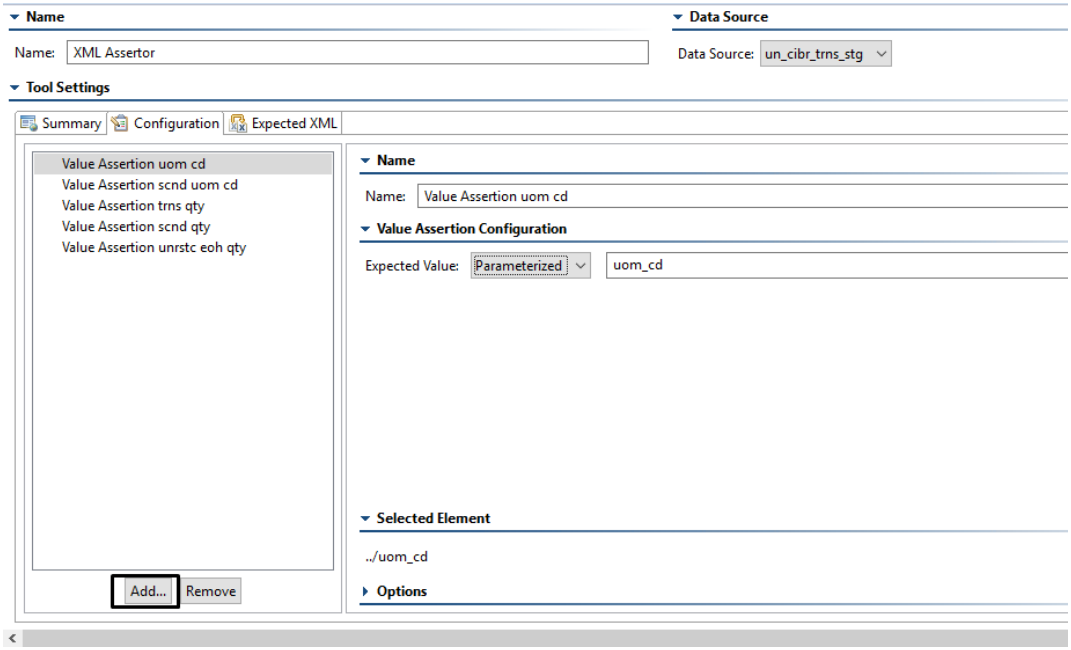
12. Run the Test Suite

a. Click Test Suite

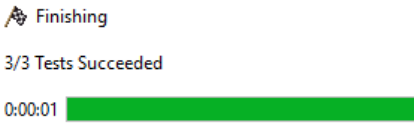
b. Click play button 

13. Add Output
    a. Right click test suite, add Output
    b. Select XML Assertor. Click Finish
    c. Add value assertions for all columns you need to validate in the table.





14. Test it
    a. Run it again. Check Results



13 Best Practices for SOATest

13.1.1 Refresh and Save BKM

After every test and periodically the tool works better if you REFRESH REFRESH REFRESH. This can be found in two places, under “File” on the menu or the ICON with the two arrows upper left. This is very important after you run tests as the Data banks and assertions get cached.

Save or Save ALL often. This can be found in two places, menu under ‘File” or the ICON

If refreshing and saving do not work sometimes you will have to shut down SOATest and restart.

Anexo 27 - Resultados de la Encuesta

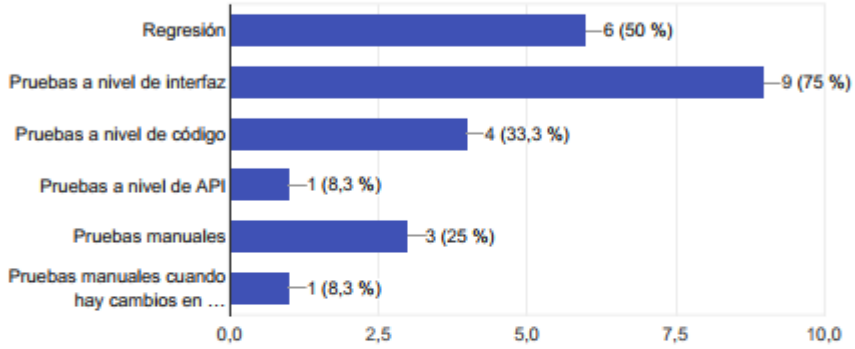
Proceso de pruebas manuales en ICOST

12 respuestas

Publicar datos de análisis

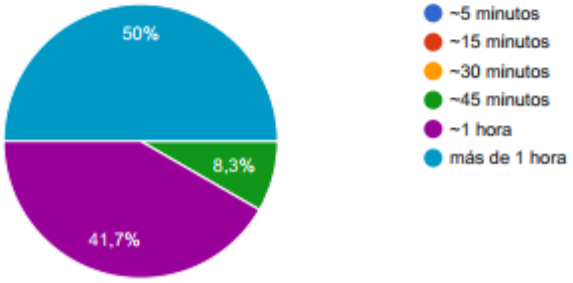
Qué tipos de pruebas se realizan en ICOST?

12 respuestas



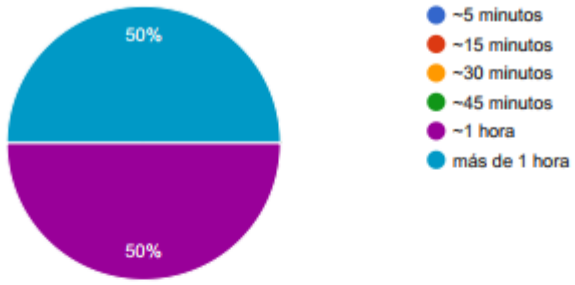
Cuánto tiempo tarda en probar un screen en ICOST?

12 respuestas



Cuánto tiempo tarda en probar un reporte de SSRS en ICOST?

12 respuestas



Cómo prueba el código que desarrolla?

11 respuestas

Lo reviso visualmente y viendo los datos que me retorna el store procedure

Hago peer-programming con compañeros del equipo y comprobamos la correcta funcionalidad

Visualmente y hago mis propias pruebas en el mismo código

Manualmente en SQL

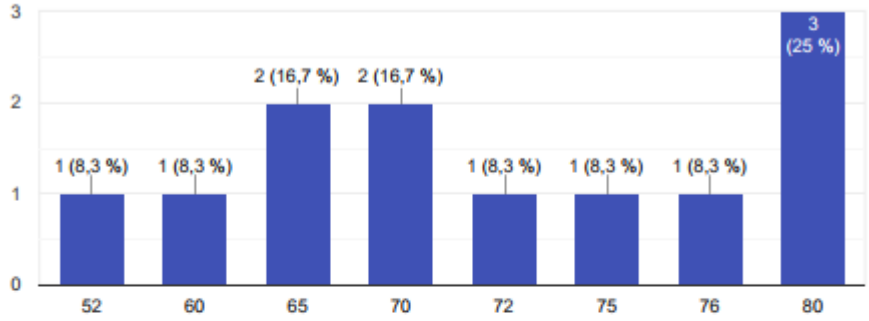
Lo reviso con algún compañero para la comprobación de lo que hice es correcto

No realizamos pruebas unitarias automatizadas, todo se realiza manualmente con código customizado SQL o visualmente se revisa los datos que retorna el procedimiento almacenado

Reviso los criterios de aceptación del cambio para comprobar que se cumple con lo solicitado

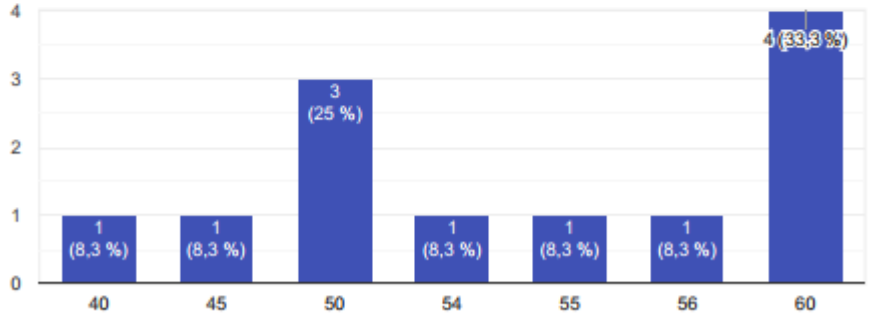
Cuántas pantallas tiene ICOST? un aproximado

12 respuestas



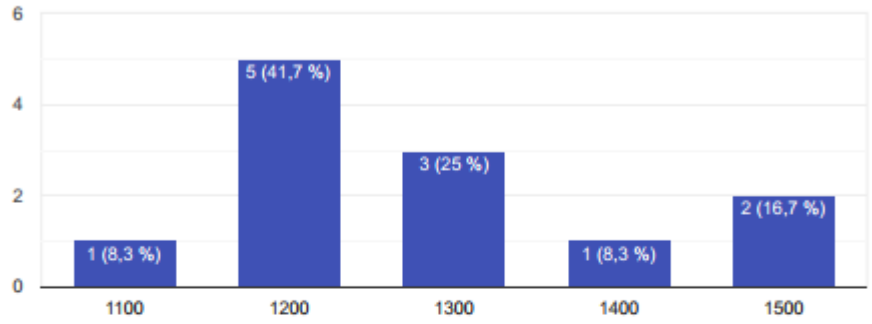
Cuántos reportes SSRS tiene ICOST? un aproximado

12 respuestas



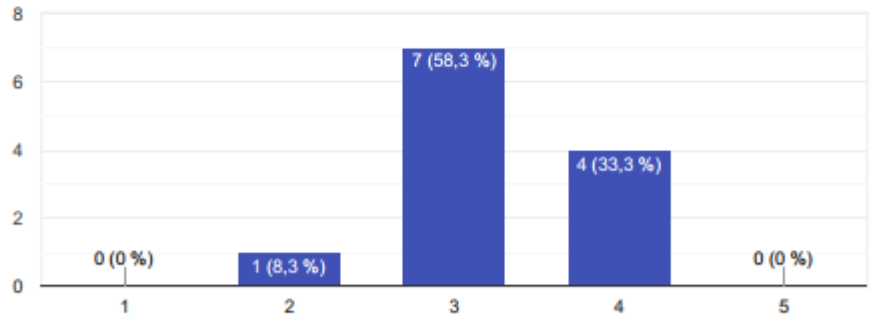
Cuántos procedimientos almacenados tiene ICOST? un aproximado

12 respuestas



En general, cómo ve la calidad de ICOST?

12 respuestas



Anexo 28 – Gráfico Marco de Trabajo