Diseño de interfaz de usuario para el entrenamiento de las actividades de perforación de pozos petrolíferos verticales

Angélica Mercedes Basto García, Laura Vanessa López Serrano

Trabajo de Grado para Optar el título de Ingeniero de Sistemas

Director

Gabriel Rodrigo Pedraza Ferreira

Doctor en Informática

Codirector

Luis Eduardo Bautista Rojas

Magíster en Ingeniería de Sistemas e Informática

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela de Ingeniería de Sistemas e Informática

Bucaramanga

2019

Dedicatoria

A mi mamá Eilen Edith Serrano de quien he aprendido su espíritu luchador y su constancia, a mi papá Holguer Mauricio López quien siempre ha tenido una palabra de aliento y sabiduría, a mi hermano Daniel López por ser siempre mi amigo y mi compañía.

A mi abuela Raquel Ríos y a mí Nana Liliana quienes sé que siempre me tienen en sus oraciones y de quienes he aprendido a salir adelante a lo largo de mi vida.

A mi familia quienes siempre han sido mi apoyo incondicional y de quienes siempre estaré enormemente agradecida.

A mi compañera de trabajo de grado Angélica Basto, por acompañarme en este arduo camino y con quien superamos juntas diferentes obstáculos.

A mi novio Aron Forero por acompañarme y apoyarme a lo largo de toda mi carrera universitaria.

A Dios, a la vida y a mis ángeles por ser tan buenos conmigo.

Laura Vanessa López Serrano

Agradecimientos

Gracias Dios por permitirme vivir este capítulo y culminarlo con éxito bajo tu voluntad. Gracias por los grandes padres que me regalaste, padres todo amor, comprensión, guía y sensatez.

Gracias por todas aquellas personas que pusiste en mi camino y que fueron de gran apoyo en el desarrollo de este proyecto; especialmente, mis hermanos Laura y Fabián, mi compañera de proyecto de grado Laura López Serrano, mi novio Ángel Valbuena, demás familiares y amigos.

Angélica Mercedes Basto García.

Agradecemos a los profesores Gabriel Pedraza Ferreira y Luis Eduardo Bautista Rojas por su entrega, colaboración y guía en el desarrollo del proyecto.

Agradecemos al grupo de investigación CAGE y SC3 por brindarnos sus instalaciones y a nuestros compañeros especialmente a Sergio, Pablo, Jonathan y Diego por su apoyo y ayuda en el desarrollo de nuestro proyecto.

Agradecemos a los creadores del simulador SSAPP por permitirnos tomar como base su proyecto para la creación del nuestro, además de brindarnos su ayuda, explicación y apoyo en este proceso.

Contenido

Pág.

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Lista de Apéndices

(Ver apéndices adjuntos en el CD y pueden visualizarlos en la Base de Datos de la Biblioteca UIS)

Resumen

Título: Diseño de interfaz de usuario para el entrenamiento de las actividades de perforación de pozos petrolíferos verticales.[*]

Autores: Laura Vanessa López Serrano, Angélica Mercedes Basto García.[**]

Palabras clave: Simulación, perforación de pozos, problemas operacionales, toma de decisiones, entrenamiento, información procedimental, MBUID.

Descripción:
Actualmente uno de los tipos de *software* más sobresalientes del mundo de la informática son los interactivos de simulación, ya que hacen parte de las nuevas tecnologías y se pueden implementar en diferentes industrias.

Hoy por hoy, la simulación es aplicable en diversos campos y es de gran utilidad como herramienta de enseñanza y aprendizaje, un ejemplo de esto son los simuladores para la industria petrolera, ya que, durante las operaciones de perforación de pozos, es de gran importancia que los operarios cuenten con la experiencia suficiente para la correcta toma de decisiones en la ejecución de las actividades. Por esta razón se han implementado simuladores en la industria petrolera que permiten capacitar a los aprendices, permitiendo que se familiaricen con los controles, los procesos operacionales y el manejo de los posibles problemas operacionales que se presenten durante la perforación.

En base a lo anterior, este proyecto está orientado a la creación de una Interfaz de usuario tutor integrada a un entorno de simulación de perforación de pozos petrolíferos. Dicha interfaz pretende entrenar al operador en la correcta toma de decisiones a la hora de dar solución a un problema operacional, haciendo uso de información procedimental. Este tipo de información permite definir el conjunto de actividades que el aprendiz debe llevar a cabo para dominar la técnica o habilidad de solución del problema operacional presentado, para posteriormente poner en práctica los conocimientos adquiridos en un ambiente real. El desarrollo de esta interfaz se basa en el enfoque Model Based User Interface Development – MBUID, pretendiendo que tanto su diseño como contenido sea flexible de tal forma que permita al diseñador mejorar dicha interfaz o recrear la misma a medida que el producto vaya madurando.

[*] Trabajo de grado

[**] Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas e Informática Director: Gabriel Rodrigo Pedraza Ferreira, Doctor en Informática. Codirector: Luis Eduardo Bautista Rojas, Magíster en Ingeniería de Sistemas e Informática.

Abstract

Title: User interface design for training vertical oil well drilling activities.*

Authors: Laura Vanessa López Serrano, Angélica Mercedes Basto García.**

Keywords: Simulation, Well drilling, Operational Issues, Decision-making, Training, Procedural Information, MBUID.

Description:
Nowadays, one of the more outstanding types of software in the world of informatics are the interactive simulation software, since they are part of the new technologies and they can be implemented in different types of industries.

At the present time, the simulation is applicable in several fields and has got a great utility as an educational tool, one example of this are the simulators for the petroleum industry, because during the well drilling operations, it is of great importance that the operators counts with the enough experience for the correct decision making in the execution of the activities. For this reason, simulators have been implemented in the petroleum industry to train apprentices, allowing them to get familiar with the controls, the operational procedures and the management of the possible operational issues that could arise during drilling.

Based on the above, this project is aimed to the creation of a user interface tutor integrated to a petroleum well drilling simulation environment. This interface aims to train the operator on the correct decision making at the time of give a solution to an operational issue, using the procedural information. This type of information allows you to define the set of activities that the apprentice should do to dominate the technique or ability to solve the presented operational issue, then put the acquired knowledge into practice in a real environment. The development of this interface is based on the Model Based User Interface Development (MBUID) approach, pretending that both its design and content to be flexible in such a way that allows the designer to improve that interface or recreate the same as the product get mature.

* Degree work

** Faculty of Physicomechanical Engineering. School of Systems and Computer Engineering Director: Gabriel Rodrigo Pedraza Ferreira, Doctor in Computer Science. Co-director: Luis Eduardo Bautista Rojas, Master in Systems and Computer Engineering

Introducción

La actividad de perforación de pozos de petróleo o gas es una de las principales actividades de la industria petrolera, puesto que es la única manera de verificar si hay petróleo en el sitio estudiado con anterioridad. La perforación de pozos petroleros presenta un gran número de tareas de alto riesgo tanto para los operarios y el proceso como para el medio ambiente; por esta razón, las compañías petroleras y universidades invierten considerable tiempo, dinero y recursos en las capacitaciones técnicas y practicas necesarias para proveer el personal calificado.

Actualmente el apoyo que la tecnología y el avance tecnológico brindan a las diferentes industrias es incalculable, particularmente la industria petrolera en el área de perforación de pozos petrolíferos, se ve altamente beneficiada al hacer uso de tecnologías como la simulación para el entrenamiento del personal, puesto que disminuye los recursos económicos y el tiempo que estos proyectos requieren, además de la disminución de los diversos errores humanos que se pueden presentar dado a que esto se ejecuta desde un ambiente seguro, donde se permite el error sin generar pérdidas considerables.

Previo al desarrollo de este proyecto se han hecho estudios de los distintos problemas operacionales que se presentan durante la perforación de pozos petroleros, además de un diseño técnico y pedagógico; esto se muestra en el proyecto titulado “Sistema para la simulación de las actividades de perforación de pozos petrolíferos verticales”, desarrollado por estudiantes de la escuela de Ingeniería de Petróleos de la Universidad Industrial de Santander; en este proyecto, se realizó el desarrollo de un ambiente virtual que simula el ambiente en el cual se lleva a cabo el proceso de perforación.

Durante el entrenamiento es importante brindar al aprendiz las tareas necesarias para llevar a cabo un objetivo, de tal forma que facilite la realización del proceso de manera óptima; es por esto que se hace beneficioso el uso de información procedimental logrando que el aprendiz no solo comprenda la información brindada, sino que también adquiera las habilidades necesarias en la toma de decisiones y así poner en práctica los nuevos conocimientos en un ambiente real.(León, 2012)

Por esta razón, este proyecto pretende diseñar e integrar al ambiente virtual desarrollado en el proyecto mencionado anteriormente, un prototipo de interfaz de usuario, que brinde al perforador la información procedimental necesaria para solucionar los problemas operacionales presentados.

La elaboración de un producto software de este tipo requiere de un alto esfuerzo puesto que es necesario llevara a cabo una minuciosa investigación al problema que se desea solucionar; así como al entorno en el cual fue desarrollado dicho proyecto, permitiendo la correcta creación de los modelos y la explotación de información recopilada en los mismos; de tal forma, que se permita al diseñador producir diferentes UI's a medida que el producto vaya madurando; a su vez, esta interfaz de usuario debe ser visualmente atractiva, flexible y fácil de usar garantizando una buena calidad de la misma. ("Un buen diseño de la interfaz de usuario en el éxito de aplicaciones de software", s/f)

1. Objetivos

1.1 Objetivo General

Diseñar y codificar un prototipo de interfaz de usuario Tutor para el entrenamiento en las actividades de toma de decisiones en la perforación de pozos petrolíferos verticales.

1.2 Objetivos Específicos

- Definir las actividades de toma de decisiones que se van a desarrollar en la solución al problema operacional presentado.
- Aplicar un enfoque de desarrollo basado en modelos MBUID (Model Based User Interface Development) para el diseño de la interfaz de usuario tutor.
- Codificar un módulo de entrenamiento en forma de tutor usando el entorno de desarrollo Unity y el lenguaje de programación C#.
- Diseñar y ejecutar un plan de pruebas técnicas de software (unitarias y de integración).

2. Cumplimiento de Objetivos

Tabla 1. *Cumplimiento de Objetivos.*

Objetivo	Descripción	Capitulo	Pág.
Definir las actividades de toma de decisiones que se van a desarrollar en la solución al problema operacional presentado.	Por medio de información procedimental algorítmica y de los modelos instruccionales, se definieron las actividades de toma de decisión necesarias para dar solución al problema operacional pega por empaquetamiento.	Capítulo 4 sección 4.1	**43**
Aplicar un enfoque de desarrollo basado en modelos MBUID (Model Based User Interface Development) para el diseño de la interfaz de usuario tutor.	Se crearon los modelos de tarea para aplicar el enfoque MBUID, así mismo se crearon las reglas de concretización necesarias para realizar la transformación de un modelo de alto nivel de abstracción a otro de menor nivel de abstracción como lo expone dicho enfoque.	Capítulo 4 sección 4.5	**51**
Codificar un módulo de entrenamiento en forma de tutor usando el entorno de desarrollo Unity y el lenguaje de programación C#.	Se realiza una descripción del procedimiento de codificación e integración del diseño y contenido de la interfaz de usuario Tutor al entorno Unity.	Capítulo 4 sección 4.6	**80**
Diseñar y ejecutar un plan de pruebas técnicas de software (unitarias y de integración).	Se aplicó un plan de pruebas manuales, unitarias y de integración, mediante las cuales se identificó el porcentaje de efectividad en la aplicación del enfoque MBUID.	Capítulo 4 sección 4.7	**84**

3. Marco referencial

3.1 Perforación de pozos y problemas operacionales durante la perforación.

La perforación de pozos es una actividad basada en la perforación del subsuelo con el fin de extraer hidrocarburo, siendo este la principal fuente de energía en todo el mundo. La perforación tiene como objetivo, construir un pozo que permita la conexión del yacimiento con la superficie para así, explotarlo de forma segura y al menor costo posible para posteriormente tratarlo y comercializarlo.(Quijada, Caraballo, & Garcias, 2011)

De acuerdo a la profundidad a la que se encuentre el yacimiento (generalmente 3000 o 4000 metros), las formaciones que se van a atravesar y las condiciones propias del subsuelo, se selecciona el equipo de perforación más indicado; debido a que la perforación de pozos demanda mucho tiempo y recursos económicos, es necesario que el equipo de perforación instale e inicie la perforación una vez los geólogos y geofísicos han acordado que el lugar es apto para la búsqueda de hidrocarburos (“Perforación”, s/f).

Además del equipo de perforación, también se cuenta con elementos auxiliares como: la tubería, las bombas, los tanques, un sistema de seguridad que consiste en válvulas de cierre del pozo para su control y operaciones de rutina, y los generadores eléctricos de distinta capacidad según el tipo de equipos. Durante el proceso de perforación de pozos es común que se presenten problemas operacionales, los cuales pueden ser causados tanto por fallas técnicas en el funcionamiento de los diferentes equipos usados en el proceso, como por fallas geológicas.(Quijada et al., 2011)

La presencia de estos problemas operacionales afecta el curso normal del proceso de perforación, generando pérdidas para la industria petrolera si el problema no es atendido según lo establecido, y por el personal con la suficiente experiencia para brindar el mejor desenvolvimiento a la situación.

Entre los problemas más frecuentes y críticos se encuentran la pérdida de circulación o perdida de fluido hacia la formación, aunque una pérdida parcial no necesariamente impide continuar la perforación. Se pueden tener grandes pérdidas si la pérdida aumenta constantemente o se pierde completamente la circulación ("Pérdida de Circulación (Ocurrencia, Detección, Prevención y Soluciones)", s/f).

Otro problema frecuente es el conocido como pega por empaquetamiento, se presenta cuando partículas pequeñas como recortes de la formación o desechos caen dentro del pozo, esto generalmente se presenta al usar herramientas con diámetro cercano al del pozo, puesto que las partículas se acumulan alrededor de la herramienta bloqueando el espacio anular entre la tubería, esto genera la pega de la tubería debido a pega por empaquetamiento ("Problemas comunes de perforación relacionados con (1)", s/f).

Otro problema operacional muy común es el influjo. Algunas de las causas de que este problema se presente son:

- No tener un peso de lodo adecuado para contrarrestar la presión de la formación; es decir la presión de poro es menor que la presión hidrostática del pozo, ya que los fluidos siempre fluirán en la dirección de la presión menor.(Toinga, 2012)

- Presencia de presiones dinámicas y migratorias, esto generalmente se debe al movimiento de la broca en lugares donde el pozo experimenta presiones más bajas que las de la formación.(Toinga, 2012)

3.2 Simulación

Los simuladores son programas informáticos, los cuales permiten reproducir: sensaciones, experiencias y la realidad de un sistema; son las tecnologías más avanzadas para entrenar operarios en situaciones de riesgo, evaluar el nivel de conocimientos y cumplimiento de las normas de seguridad, y muchas más actividades sin arriesgar vidas humanas, daños en maquinaria o medio ambiente. En conclusión, un simulador permite realizar todo aquello que no se puede hacer en una máquina o ambiente real por motivos de seguridad tanto ambiental como humana, además de tener grandes beneficios económicos.(“Descripción General | e-Tech Simulation”, s/f)

Las diferentes industrias desde hace varios años atrás hacen uso de la simulación para el entrenamiento y la capacitación, tanto del personal aprendiz como del operario experimentado, ya que esto permite el perfeccionamiento del personal y la constante actualización de sus conocimientos caminando de la mano con los avances tecnológicos que se presentan diariamente, obteniendo así la capacitación continua y evitar situaciones de alto riesgo. (“Descripción General | e-Tech Simulation”, s/f)

3.3 Información procedimental.

La información procedimental o también llamado contenido procedimental, constituye en un conjunto de acciones organizadas y ordenadas, cuyo fin es facilitar el alcance de un fin propuesto; estas acciones son realizadas reiterativamente por el aprendiz llevándolo a dominar la técnica o habilidad, además de desarrollar su capacidad en el "saber hacer" y la toma de decisiones.(*Contenidos conceptuales, procedimentales y actitudinales*, 2009)

El fin del desarrollo de las acciones anteriormente mencionadas, no es sólo la comprensión de los enunciados, las fórmulas, las reglas de actuación y las instrucciones, sino también implica que el aprendiz adquiera las habilidades mentales que le permitan establecer criterios para poner en práctica los nuevos conocimientos.

Se tienen tres tipos de información procedimental, los cuales son expuestos a continuación:

- Información general: La información o contenido procedimental enfocado al proceso de búsqueda de información, comprensión de la información obtenida y comunicación de dicha información ya sea de forma escrita u oral.
- Información algorítmica: Esta información o contenido establece tanto el orden como el número de condiciones que han de cumplirse a fin de solucionar un problema. Siempre que la ejecución de estos pasos se realice en el orden previamente establecido se obtendrán resultados idénticos.
- Información heurística: Esta información o contenido es diseñada en función del entorno; es decir su aplicación no se realiza de manera automática ni bajo el mismo patrón al momento de dar solución de un problema. (León, 2012)

3.4 Modelos Instruccionales

El diseño instruccional es un proceso en donde se planean, diseñan, implementan y evalúan contenidos educativos en conjunto con el uso de la tecnología como herramienta, conforman el contenido educativo digital, el cual la principal finalidad es crear experiencias de aprendizaje mejorando la calidad de este para que el alumno adquiera conocimiento. Este método da valor al pedagogo para diseñar los materiales y estrategias didácticas del curso. (Belloch, 2012)

3.4.1 Diseño de modelos instruccionales. El planteamiento de un diseño instruccional evalúa su estructura y deberá responder a algunas preguntas como:

- ¿Qué objetivo de aprendizaje se propone?
- ¿Qué información se organizará?, como se organizará y presentará esa información?
- ¿Qué actividades se propondrán al alumno?

Existen diversos modelos instruccionales, por ejemplo, Robert Gagné plantea un sistema que se compone de diferentes niveles de aprendizaje y cada uno requiere diferentes tipos de instrucción, estos niveles pueden organizarse jerárquicamente dependiendo de la habilidad del aprendiz o el modelo ADDIE propone 5 fases como guía dinámica y flexible para el desarrollo de prácticas efectivas en el desempeño, terminadas estas fases el proceso se repite. (Gomez, 2018)

Es importante que en el desarrollo del diseño de modelos instruccionales, se tenga claridad en todo el proceso y guía hacia la búsqueda de soluciones (Alarcon, 2014), es por esto que existen varias teorías para la realización de un diseño instruccional, tres de las principales teorías son:

1. Constructivismo: el estudiante está en control de su propio aprendizaje, por eso es de gran importancia que se genere buenos problemas, cree actividades de aprendizaje en grupo y guie el proceso de construcción del conocimiento.

2. Cognitivismo: significa que observa nuevos patrones de comportamiento y se enfoca en como aprender, esto requiere analizar las tareas apropiadas para el procesamiento efectivo y eficiente de la información y aplicar una variedad de estrategias.

3. Conductismo: estudia comportamientos que son observables y medibles en un individuo y que llegan a ser automáticos de acuerdo con la repetición de estos. Por esto es importante la retroalimentación y orientación al aprendiz al dominio de actividades y habilidades.

3.5 Interacción Hombre Máquina y Toma de decisiones

Hoy en día no se tiene una definición concreta para el conjunto de términos interacción hombre-máquina, pero se puede decir que hace referencia a la relación dada mediante el intercambio de información entre el ser humano y los computadores (maquinas) por medio de una interfaz, permitiendo que el ser humano extienda sus capacidades. Su objetivo es que el intercambio sea cada vez más eficiente, minimizando errores, aumentando la satisfacción del usuario y haciendo más productivas las tareas que se presentan entre el ser humano y los computadores.(Pamela, s/f-b)

En la interacción hombre-máquina se encuentran involucradas múltiples disciplinas, las cuales están relacionadas con ciencias de la computación y ciencias humanas; para el buen desarrollo de la comunicación e intercambio de información entre el humano y la máquina, es necesario que las máquinas cuenten con el adecuado sistema operativo, técnicas gráficas, lenguajes de programación

y entornos de desarrollo, también se deben tener conocimientos previos en teoría de la comunicación, diseño gráfico e industrial, ciencias sociales, psicología cognitiva entre otras. (“Interaccion entre Humanos y Computadoras: Tendencias Actuales y Futuras Introduccion del Editor Invitado | Septiembre 2014 | Computing Now - IEEECS”, s/f)

Actualmente los sistemas que permiten al humano interactuar con la máquina y los procesos tienen una gran importancia en la automatización moderna ya que cada vez realizan operaciones más complejas, necesitando de interfaces para que los humanos puedan realizar la tarea de programación de manera ágil y eficiente. Con el paso del tiempo se han hecho comunes las interfaces gráficas, táctiles, basadas en entornos 3D, realidad virtual y realidad aumentada; permitiendo un control directo sobre las variables del sistema y un análisis visual del mismo. Por medio de esta interacción hombre-máquina y gracias a las capacidades gráficas de los computadores actuales, es posible simular el comportamiento de los procesos y las máquinas antes de su implementación real, además de la capacitación de los operadores.(Pamela, s/f-a)

Toma de decisiones

La toma de decisiones indica que una situación es valorada y considerada a profundidad para seleccionar un camino adecuado para seguir el proceso según las diferentes opciones y operaciones; aplicar un buen procedimiento o modelo de toma de decisiones, ahorrará tiempo, esfuerzo y energía.(“Importancia de la toma de decisiones.”, 2008)

En la toma de decisiones es de gran importancia:

- Evaluar las diferentes alternativas teniendo presente el análisis costo-beneficio y marginal.
- Tener claros los factores limitantes y determinar los factores importantes de una decisión.

- Tener la capacidad de desarrollar nuevas ideas.
- Conocer los tipos de decisiones programadas y no programadas.
- Usar métodos actualizados y modernos para la toma de decisiones.

Los siguientes pasos ayudan a tener éxito en la toma de decisiones:

- Describir el problema o el objetivo al que queremos llegar. En necesario tener bien definido el problema y unos objetivos claros, para así tomar las decisiones que podrían ayudar en su solución.
- Análisis del problema o la decisión a tomar. En este paso se analizará el problema de forma completa y objetiva, para esto es necesario recolectar toda la información posible tomando siempre la información relevante.
- Generar opciones. Se analizan todas las posibilidades y sus respectivas consecuencias, estudiando si estas posibilidades permiten alcanzar los objetivos establecidos en la fase inicial, es conveniente que sean varias personas trabajando en esta fase, pues así se tendrán más aspectos en cuenta.
- Selección de la alternativa que finalmente llevaremos a cabo. La alternativa seleccionada o decisión tomada, tendrá sus respectivas consecuencias, por lo cual se debe estar en la capacidad de asumirlas, permitiendo la mejora en los futuros procesos.
- Pasar de la decisión a la acción. Es importante no tomar decisiones en base a las emociones, por esta razón, es necesario meditar la decisión a tomar y ponerla en práctica.
- Valorar las consecuencias y el éxito de la decisión adoptada. Se debe saber rectificar si se está tomando una decisión errónea, la valoración de las consecuencias y del alcance de los objetivos permiten el perfeccionamiento de futuras decisiones y la mejora de los procesos. Es muy

importante valorar los resultados de la decisión tomada, pues sin éste paso el proceso toma de decisiones sería incompleto.(Martínez, 2014)

3.6 Diseño de la interfaz de usuario.

La interfaz de usuarios se puede definir como el medio por el cual el usuario y la maquina o dispositivo se comunican, generando así la interacción hombre máquina una de las áreas de investigación más estudiadas en los últimos tiempos; la interfaz de usuario tiene entre sus objetivos principales establecer la comunicación entre el usuario y el maquina o dispositivo, y una vez logrado este, dicha interfaz debe ser amigable e intuitiva para el usuario estableciendo una comunicación clara y facilitando su interacción (“Interfaz de usuario - EcuRed”, s/f) (Fernandez Ruiz, Angós Ullate, & Salvador Olivan, 2001).

Constantemente se tienden a confundir las definiciones interfaz de usuario (UI) y experiencia de usuario, siendo estas completamente diferentes, pero siempre una ligada a la otra; la interfaz de usuario hace referencia a la interfaz visual de una herramienta de software mientras que la experiencia de usuario trata la experiencia que tiene el usuario al interactuar en la interfaz gráfica.(Ramírez-Acosta, s/f)

3.7 Desarrollo de interfaz de usuario basada en modelos (MBUID)

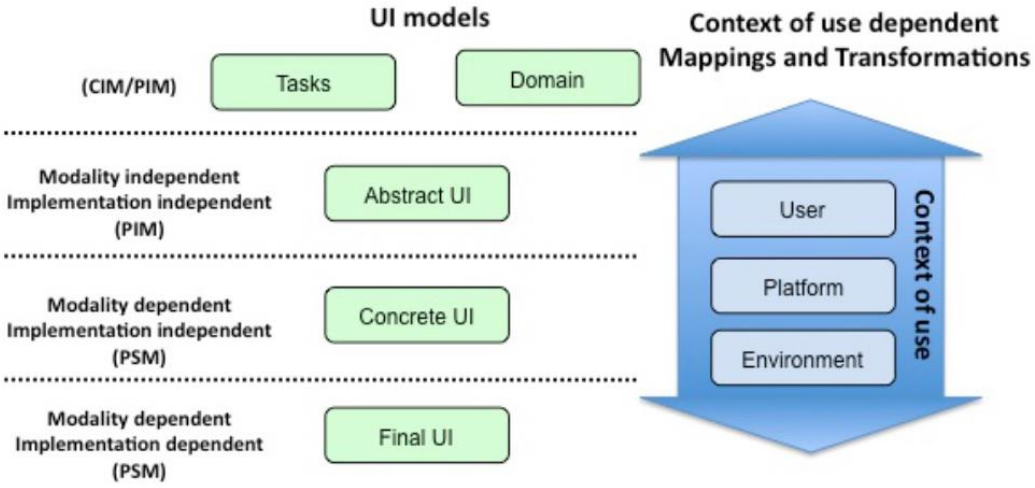


Figura 1. Asignación y Transformación Entre Niveles de Abstracción Según el Contexto de Uso. Adaptado de W3C. Introduction to Model-Based User Interfaces. Tomado de https://www.w3.org/2011/mbui/drafts/mbui-intro/

Model-Based User Interface Development (MBUID) en español Desarrollo de Interfaz de Usuario Basada en Modelos, es un paradigma metodológico de desarrollo de software centrado en la creación y explotación de modelos de dominio y no en algoritmos, tiene como objetivo principal disminuir el esfuerzo en el desarrollo de interfaces de usuario explotando toda la información recopilada a través de modelos declarativos para ayudar a los diseñadores a producir diferentes IUs.

Desde 1990 se ha venido trabajando con el enfoque MBUID, en un principio MBUID se centró principalmente en la identificación y la abstracción de los aspectos relevantes de una interfaz de usuario, luego se centró en la extensión del modelo de UI, dando paso a una tercera generación la cual estuvo marcada por la gran cantidad de nuevas plataformas y dispositivos de interacción, por

lo cual los desarrolladores y diseñadores tuvieron que desarrollar una UI para diferentes dispositivos con diferentes características, haciendo de esto un reto; finalmente nos encontrarnos en la cuarta generación, actualmente MBUID se centra en el desarrollo de UI sensibles al contexto para una gran variedad de plataformas, dispositivos y modalidades además de la integración de aplicaciones web. Hoy en día, este enfoque se conoce como desarrollo dirigido por modelos (model-driven development). Algunos ejemplos son CTT + MARIA, UsiXML, useML + DISL+UIML. (Jaime et al., 2014)

Actualmente, la comunidad de ingenieros de HCI acepta ampliamente y toma como referencia para estructurar y clasificar procesos de desarrollo de interfaz de usuario basado en modelos, al proyecto Cameleon Reference Framework (CRF), ya que este proyecto descompone el diseño de la interfaz de usuario en un número de diferentes componentes médiate los cuales se busca reducir el esfuerzo en la orientación de múltiples contextos de uso. Este es un proyecto financiado por la Unión Europea.

Como se observa en la figura 1, Cameleon especifica que para que una herramienta de IU pueda ser considerada un entorno de desarrollo de IU basado en modelos es necesario que cuente con un modelo de alto nivel, un modelo de nivel abstracto y un modelo de tarea o un modelo de dominio o ambos; además de tener una relación clara y automatizada entre el modelo anterior y la interfaz de usuario operativa deseada; es decir, debe existir algún tipo de transformación automática ya sea de generación basada en conocimiento o una simple compilación para implementar la IU final. (Bottoni et al., 2013)

3.7.1 Modelo de tareas de usuario. El modelo de tareas es parte fundamental en el desarrollo de interfaces de usuario basada en modelos puesto que permite la identificación y descripción de las tareas que podrá realizar el usuario a través de la interfaz de usuario, así como la especificación de las relaciones temporales que existen entre dichas tareas.

El modelo de tareas se diseña mediante el análisis de los casos de uso realizados basados en los requisitos especificados por el cliente, donde cada caso de uso representa una tarea abstracta que el usuario debe llevar a cabo. Este modelo de tareas puede ser llevado a cabo ya sea mediante métodos textuales basados en análisis cognitivo como GOMS (Goals, Operators, Mathods, Selection rules) o métodos basados en formalismos como CTT (ConcurTaskTrees).(Muñoz Márquez, 2007)

Las siguientes son algunas características deseables en las técnicas de modelado:

- Contar con una estructura jerárquica que permita manejar distintos niveles de abstracción y de lugar a un refinamiento progresivo del análisis de tareas.
- Permitir de manera sencilla la creación de relaciones temporales entre las tareas.
- Fácil comprensión tanto para los usuarios como para los diseñadores permitiendo una comunicación con mayor claridad entre ambas partes.
- Existencia de una herramienta automática que permita realizar el modelado de las tareas.

3.7.2 Modelo de interfaz de usuario abstracta (AUI). El modelo de interfaz abstracta modela información tanto independiente de la implementación como independiente de la plataforma o tecnología a usar; este modelo está compuesto por objetos de interacción abstracta (AIO) los cuales se dividen en dos tipos: componentes de interacción abstractos (AIC) y contenedores abstractos (AC). (Muñoz Márquez, 2007)

- **Componente de interacción abstracto (AIC)**: Un componente de interacción abstracto (AIC) tiene cuatro tipos de facetas asociadas, estas describen las capacidades de interacción de los objetos abstractos; a continuación, se describen cada una de ellas.
    - **Faceta de entrada:** describe una acción de entrada soportada por un componente de interacción abstracto (AIC).
    - **Faceta de salida:** describe los datos que pueden ser mostrados al usuario por el componente de interacción abstracto (AIC).
    - **Faceta de navegación:** describe la posibilidad de transición del componente.
    - **Faceta de control:** describe el enlace entre un componente de interacción abstracto (AIC) y las funciones del sistema.
- **Contenedor abstracto (AC):** Es una entidad que permite la agrupación de manera lógica de otros contenedores o componentes abstractos, además esta entidad soporta la ejecución de un conjunto lógico/semántico de tareas conectadas; los contenedores abstractos no pueden tener facetas asociadas. (Muñoz Márquez, 2007)

3.7.3 Modelo de interfaz usuario concreta (CUI). Este modelo representa la IU en un nivel de abstracción menor que el usado en la IU abstracta. La especificación de esta interfaz es independiente de la tecnología en la que será implementada la interfaz más no de la modalidad que será usada. (Abraham, 2007)

La interfaz de usuario concreta es un modelo específico de interfaz de usuario que permite la especificación de la apariencia y el comportamiento de una interfaz de usuario con elementos que pueden ser percibidos por los usuarios

El modelo CUI está conformado por Objetos de interacción concretos (CIO) y relaciones concretas más conocido como Objetos gráficos y relaciones y objetos auditivos y relaciones. Un CIO se define como una entidad que los usuarios pueden percibir y / o manipular. El control de diálogo definido en el modelo abstracto IU permite un flujo de control entre los objetos de interacción concretos.

En esta etapa se relacionan los elementos correspondientes del modelo de tareas con el conjunto de elementos de la interfaz abstracta. (Muñoz Márquez, 2007)

3.7.4 Modelo de interfaz usuario final (FUI). Una vez tomadas una serie de decisiones de diseño sobre la modalidad y la tecnología a utilizar se aborda la fase final del desarrollo.

Una FUI puede representarse en cualquier lenguaje de programación de UI (por ejemplo, el kit de herramientas de Java UI) o en lenguaje de marcado (por ejemplo, HTML).

Por lo tanto, este modelo representa el código que será ejecutado en la plataforma definida desplegando la interfaz de usuario; por esta razón, este modelo de interfaz de usuario es independiente tanto de la plataforma en la que se ejecutara la interfaz como de la modalidad usada para interactuar con la plataforma.

Una vez obtenido el modelo de interfaz concreto es posible relacionar los elementos a una plataforma dependiente según los requerimientos específicos obtenidos en el proceso y las actividades. (Abraham, 2007)

3.7.5 Relación entre modelos. Según el artículo "Introducción a UI basado en Modelos" (Bottoni et al., 2013), para aplicar el enfoque de Cameleon, en el cual se hace necesario el conjunto de modelos de UI (modelo de tareas, modelo de UI abstracto, modelo de UI concreto y modelo de UI final) y hacer la transición de uno a otro, se hace necesario el uso de reglas de transformación, estas reglas pueden definirse como el vocabulario mediante el cual se relacionan los modelos y se puede realizar la transformación de un modelo a otro.

Las relaciones entre los modelos propuestos por Cameleon se definen como reglas de concretización, reglas de abstracción, reglas de traducción y reglas de reflexión.

Las reglas de concretización son reglas mediante las cuales se transforma un modelo particular en otro de un nivel inferior de abstracción, hasta que se alcanza el código ejecutable / interpretable. Cameleon muestra un proceso de concretización de cuatro pasos: el nivel de Tarea o Dominio se "concreta" en un modelo de UI abstracta, que a su vez conduce a una IU concreta. Una interfaz de usuario concreta se convierte en una interfaz de usuario final, generalmente mediante técnicas de generación de código; como se mencionó anteriormente, este código puede representarse en cualquier lenguaje de programación de UI o en lenguaje de marcado como lo es HTML, XML, entre otros.

Las reglas de abstracción transforman una representación de IU de cualquier nivel de abstracción a un nivel superior de abstracción; un claro ejemplo de esto es la ingeniería inversa de las interfaces de usuario.

Las reglas de traducción transforman una descripción destinada a un contexto particular de uso en una descripción al mismo nivel de abstracción, pero dirigida a un contexto de uso diferente.

Las reglas de reflexión transforman un modelo en otro en el mismo nivel de abstracción para el mismo contexto de uso (a diferencia de los diferentes contextos de uso como para la traducción).

3.8 Herramientas

3.8.1 Concur Task Tree (CTT). CTT es una notación desarrollada por Fabio Paternó utilizada para modelar las tareas que un usuario puede llevar a cabo en una aplicación interactiva. (abialeba, 2014)

La notación CTT es una representación gráfica mediante el desglose jerárquico de tareas en subtareas con un número indefinido de niveles, donde dichas tareas y subtareas pueden tener relaciones temporales entre pares de tareas de un mismo nivel, permitiendo llevar una secuencia de acciones realizadas (Tareas) por el usuario a la hora de interactuar con una aplicación para alcanzar un objetivo final; a continuación, se muestran en la Tabla 2 los diferentes operadores temporales que ofrece la notación CTT.

Tabla 2**.** *Operadores Temporales Definidos en la Notación CTT.*

Operador Temporal	**Notación**	**Descripción**
Entrelazado (Concurrencia Independiente)	T1 | | | T2	Las acciones de las dos tareas pueden realizarse en cualquier momento
Elección	T1 [ ] T2	Selección alternativa entre dos tareas. Una vez que se está realizando una de ellas la otra no está disponible hasta que termine la que está activa.
Sincronización (Concurrencia con intercambio de Información)	T1 | [ ] | T2	Las dos tareas tienen que sincronizarse en alguna de sus acciones para intercambiar información.
Desactivación	T1 [> T2	Desactivar. La primera tarea es desactivada cuando comienza la ejecución de la segunda.
Activar (Enabling)	T1 >> T2	Cuando termina la T1 se activa la T2. Las dos tareas se realizan de forma secuencial.
Activar con paso de información	T1 [ ] > > T2	Cuando termina T1 genera algún valor que se pasa a T2 antes de ser activada.
Iteración	T1*	La tarea T1 se realiza de forma repetitiva. Se estará realizando hasta que otra tarea la desactive.
Independencia de Orden	T1 | = | T2	Ambas tareas pueden ser realizadas, pero una vez comenzada una debe finalizar antes de que comenzar con la otra.
Suspend/Resume	T1 | > T2	T2 tiene la posibilidad de interrumpir a T1 que podrá ser retomada cuando aquella finalice.
Tarea Opcional	[T1]	La realización de la tarea es opcional.

Nota. Recuperado de Arribas,B (2007). coCTT: Modelado de tareas en un entorno colaborativo [Proyecto de grado]. (Blanca Azahara, 2007)

En la notación CTT se definen las cuatro categorías de tareas, estas en función del actor que la llevará a cabo; cada una de estas categorías de tarea tiene su propio tipo de tarea, este depende de la acción a desarrollar por dicha tarea; a continuación, se enseña cada categoría con sus respectivos tipos de tarea adscritos a la misma y la notación grafica con la cual se identifica cada categoría. (Blanca Azahara, 2007)

Tarea de Usuario ( ): Las tareas de usuario son aquellas realizadas completamente por el usuario, quien toma la información que recibe del entorno, estas pueden ser tareas cognitivas o físicas en las que no interfiere el sistema. Esta categoría de tareas tiene asociados los siguientes tipos: None, Comparing, Planning, y ProblemSolving.

Tarea de Aplicación ( ): Las tareas de aplicación son tareas activadas y realizadas por aplicación como lo es obtener información interna del sistema o brindar información al usuario. Esta categoría de tareas tiene asociados los siguientes tipos: None, Comparison, Feedback, GenetationAlerts, Grouping, Locate y Overview

Tarea de Interacción ( ): Estas son tareas en las que el usuario interactúa con la aplicación; por ejemplo, ingresar información al sistema, hacer solicitudes al sistema, etc. Esta categoría de tareas tiene asociados los siguientes tipos: None, SingleSelection, MultipleSelection, Edit, Control, Zooming, Filtering y DetailOnDemand.

Tarea Abstracta ( ): Son tareas con un nivel de complejidad mayor a las anteriores, por esta razón se divide en las subtareas necesarias para la realización de la tarea abstracta de forma adecuada. Esta categoría de tareas tiene asociados los siguientes tipos: None y SearchInformation.

3.8.2 C#. Visual C# es uno de los lenguajes de programación de alto nivel que pertenecen al paquete .NET, este lenguaje es actualmente uno de los lenguajes de programación más populares, ya que es un lenguaje moderno orientado a objetos que permite desarrollar una amplia gama de aplicaciones para la nueva plataforma Microsoft. Net, la cual se caracteriza por proporcionar utilidades y servicios para sacar un provecho total tanto de la informática como de las comunicaciones. Puede usar C# para crear aplicaciones cliente de Windows, servicios web XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones de base de datos, entre otros. Visual C# proporciona un editor de código avanzado, prácticos diseñadores de interfaz de usuario, un depurador integrado y herramientas que facilitan el desarrollo de aplicaciones basadas en el lenguaje C# y .NET Framework para el desarrollo de sistemas de propósito general. (“Coordinación Ojo-Mano u Óculo-Manual - Habilidad Cognitiva”, s/f)

3.8.3 Unity. Unity es un motor de videojuego multiplataforma y de código cerrado creado por Unity Technologies, que mediante un editor y scripting permite desarrollar videojuegos con un acabado profesional, su versión actual es compatible con muchas plataformas entre estas encontramos Windows, Mac, Linux, iOS, Android, Xbox, Wii, entre otras.(Candil, 2014)

Unity puede ser usado tanto por pequeñas como por grandes empresas de igual manera y es accesible al público en versión gratuita y profesional, cada una con sus ventajas y limitaciones, ofrece un entorno amigable para los programadores, artistas y diseñadores, provee de un editor

visual muy útil y completo donde fácilmente se pueden importar modelos 3D, texturas, sonidos, etc. para después trabajar con ellos. Además, incluye la herramienta de desarrollo MonoDevelop con la que es posible crear scripts en JavaScript, C# y un lenguaje de Python llamado Boo. (Aroca, 2012)

3.8.3.1 Interfaz de usuario UI en Unity: Es necesario tener en cuenta algunos conceptos básicos los objetos de diseño y componentes que Unity proporciona para crear desde cero una interfaz de usuario UI.

3.8.3.1.1 Objetos de diseño.

Canvas: Funciona como un contenedor de objetos de interfaz de usuario UI, tiene tres tipos de renderizado.

- Screen Space-Overlay
- Screen Space-Camera
- World Space

Rect Transform: Permite controlar la posición, rotación y el escalado de los elementos de interfaz de usuario (ver Tabla 3) en la escena que se está diseñando dentro del entorno *Unity*. Es una variante del *Transform* (el cual representa sólo un punto) en el que representa un rectángulo, simbolizando el área en donde se pueden encontrar sus objetos de interfaz hijos.

Tabla 3. *Propiedades del Rect Transform.*

Propiedad	Descripción
Pos (X, Y, Z)	Posición del punto de pivote del rectángulo relativo a las anclas.
Width/Height	Ancho y alto del rectángulo.
Left, Top, Right, Bottom.	Posición de los bordes del rectángulo relativo a sus anclas Se muestran en lugar de Pos y Width/Height. Cuando las anclas son separadas y forman un rectángulo.
Anchors (Anclaje)	Puntos de anclaje para la esquina inferior izquierda y la esquina superior derecha del rectángulo. Estos elementos intentarán mantener una proporción de distancia prudente en la pantalla en caso de que esta sea reescalada. Se pueden establecer Min (x, y) y Max (x, y).
Pivot	La ubicación del punto en el cual el rectángulo puede girar.
Rotation	El ángulo de rotación (en grados) alrededor del pivote y a lo largo de los ejes X, Y y Z.
Scale	El factor escala aplicado al objeto en las dimensiones X, Y y Z.

Nota. Unity Documentation. Rect Transform. Tomado de: https://docs.unity3d.com/es/current/Manual/class-RectTransform.html

Auto Layouts o diseño automático: Unity ofrece esta herramienta en caso de que el diseñador desee posicionar varios objetos dentro de un mismo grupo en donde se puede ajustar sus dimensiones y que la organización dentro de este se vea mucho más armónica y prolija, como también permite que un elemento sea diseñado de manera automática dentro de la escena. Existen 5 tipos de layouts:

- Content Size Fitter: Es el ajustador del tamaño del contenido y controla el tamaño de su propio Layout Element.

- Layout Element: Comprende un solo elemento.
- Horizontal Layout Group: Comprende un grupo de elementos que se ubican en dirección horizontal.
- Vertical Layout Group: Comprende un grupo de elementos que se ubican en dirección vertical.
- Grid Layout Group: comprende un grupo de elementos que pueden organizarse según el tamaño de estos y puede cambiar dicha organización a medida de la cantidad de los elementos que se añadan.

Color: este componente puede ser añadido a cualquier objeto UI en Unity, pero varía dependiendo del objeto que lo contenga; si es un objeto estático cómo un Panel (ver Tabla 4), tendrá solo el color de fondo de la imagen que contiene por el contrario, si es un objeto dinámico como un botón (ver Figura 2) tendrá 4 estados de color:

- Normal color: el botón no presenta ninguna acción que lo afecte.
- Highlighted color: el botón detecta el cursor.
- Pressed color: el botón es oprimido.
- Disabled color: el botón está deshabilitado.

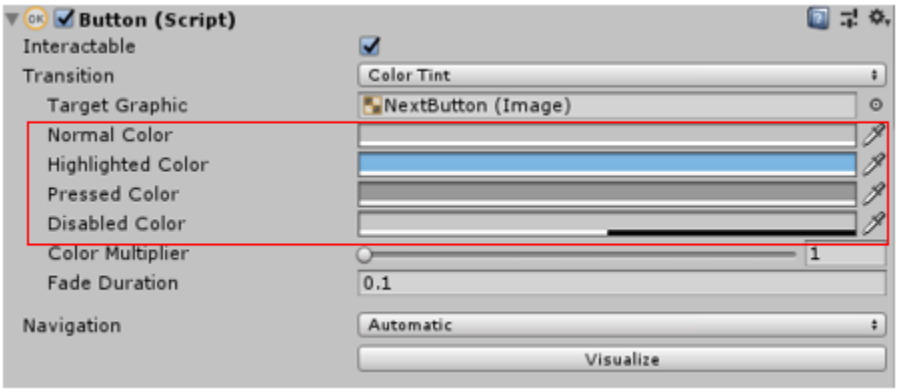


Figura 2. Presentación de Colores en el Componente de Interacción Button.

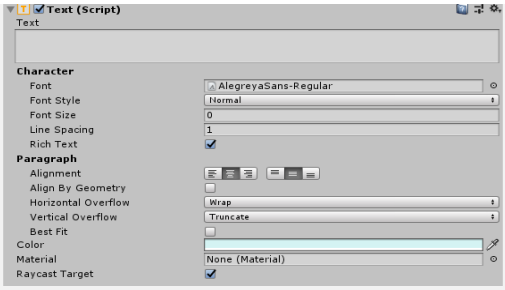
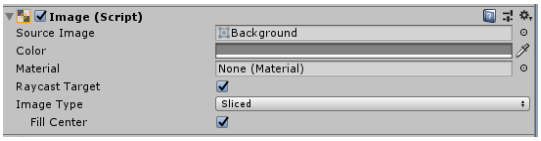
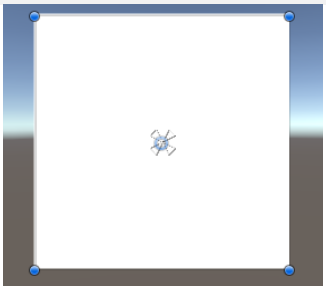
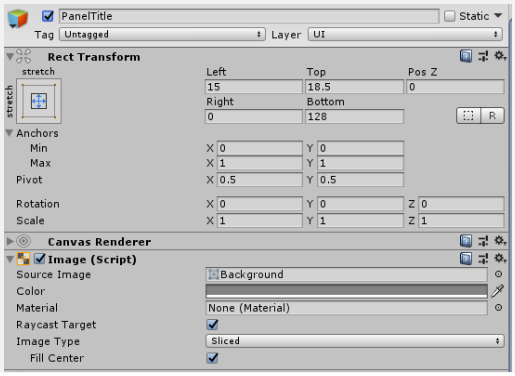
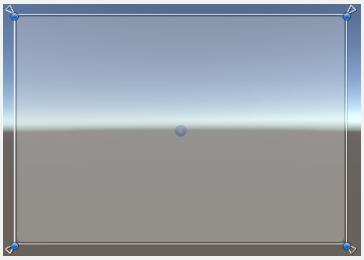
Material: se añade cuando el diseñador desea tener un efecto más realista en un objeto, usualmente se usa en objetos 3D diferentes a la interfaz de usuario. (*Materiales Unity*, 2018)

3.8.3.1.2 Componentes

Componentes Visuales:

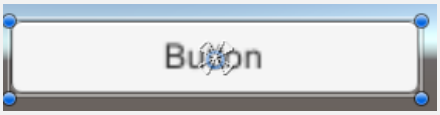
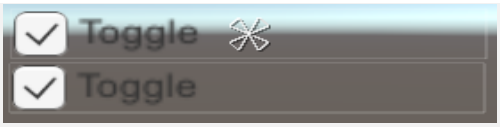
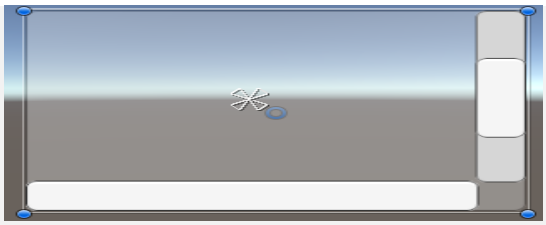
Son elementos que no determinan una función predeterminada por sí solos y corresponden a la visualización de cualquier elemento sea o no de interacción. Se especifican en la siguiente tabla:

Tabla 4. *Componentes Visuales de Interfaz de Usuario en Unity.*

Componente visual	Presentación en el inspector	Presentación en la escena
Texto		
Imagen		
Panel		

Componentes de Interacción: son aquellos componentes que realizan una acción al ser ejecutados tal como eventos del ratón, táctiles o teclado, cualquier elemento de interacción que es contenido o contiene componentes visuales con el fin de que funcionen correctamente. Estos componentes se especifican en la siguiente tabla:

Tabla 5. *Componentes de Interacción de Interfaz de Usuario en Unity*

Componente de interacción	Presentación predeterminada en la escena
Botón: Objeto de control que permite realizar acciones al pulsarlo	
Input Field: Objeto que permite el ingreso de texto.	
Toggle: Comprende un objeto o grupo de objetos de selección múltiple o con única respuesta.	
Slider: Permite al usuario seleccionar un valor numérico dentro de un rango predeterminado	
Scrollbar: Barra de Desplazamiento que permite desplazar una imagen o vista para visualizarlo completamente	
Dropdown: Lista de selección con única respuesta.	
Scrollview: Conjunto de barra de desplazamiento horizontal y vertical.	

3.8.4 Json. JSON es un formato de texto para la serialización de datos estructurados; dicho formato está basado en el subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - Diciembre 1999, así mismo es completamente independiente del lenguaje, pero se basa en convenciones ampliamente conocidas por programadores como lo son C, C++, C#, JavaScript, Perl, Python entre otros, de tal forma que Json es un lenguaje ideal a la hora de intercambiar datos dado que estas son estructuras universales soportadas por todos los lenguajes de una forma u otra. (Crockford, s/f)

JSON puede representar cuatro tipos primitivos (cadenas, números, booleanos, valores nulos) y dos tipos estructurados (objetos y arreglos). (Sanchez, s/f)

3.8.5 XML. XML es un metalenguaje desarrollado por Word Wide Web Consortium (W3C) que permite definir lenguajes de marcado adecuados a las necesidades, ejemplo de estos lenguajes lo son XHTML, MathML y SVG. ("Definición de XML - Qué es, Significado y Concepto", s/f) .

XML permite representar información estructurada como se observa en la Figura 3 de modo que pueda ser almacenada, transmitida, procesada, visualizada e impresa, por diversas aplicaciones y dispositivos. (*XML ¿QUE ES?*, s/f)

Figura 3. Estructura de un XML

Un claro ejemplo de la influencia de XML es cuando se desarrolla un programa con interfaz gráfica ya que es necesario organizar todas las imágenes de manera que se vayan cargando a medida que se necesiten, agruparlas, etiquetarlas, especificar su ubicación y relacionarlas con otros datos, según las necesidades de los diseñadores, y esto se logra haciendo uso de este metalenguaje. (“Definición de XML - Qué es, Significado y Concepto”, s/f)

4. Metodología

El proyecto se llevó a cabo en 7 fases, a continuación, se especifica lo que se realizó en cada una de ellas.

4.1 Fase 1: Investigación y documentación de la problemática a solucionar

El desarrollo del proyecto se inició con la investigación, recolección y análisis de la documentación acerca de Desarrollo de Interfaz de Usuario Basada en Modelos (MBUID) así como su diseño; así mismo se profundizó en la solución del problema operacional pega por empaquetamiento, dado que es uno de los problemas operacionales más presentados durante el proceso de perforación y el que genera más costos en la industria petrolera, además de ser el único problema operacional habilitado en el simulador UISSAPP hasta el momento, esto con el fin de definir la ventajas y limitaciones que se podrían presentar y así abordar la solución del problema de manera adecuada.

4.2 Fase 2: Capturar los requisitos de usuario.

Para el desarrollo de esta fase se llevó a cabo una reunión con el director y codirector del proyecto de manera independiente, con el fin de describir el producto final a partir del objetivo principal del proyecto, obteniendo como resultado el documento de requisitos para el desarrollo de la interfaz de usuario basada en modelos.

4.3 Fase 3: Definición del contexto PIM

En esta fase se tuvo una primera apreciación del sistema a desarrollar teniendo en cuenta el alcance del proyecto, se determinaron los actores de este y finalmente se obtuvieron tres actores, estos son: administrador, perforador y sistema.

4.4 Fase 4: Especificación de los requisitos PIM

En esta fase se tuvo una apreciación del sistema a desarrollar teniendo en cuenta el alcance del proyecto, se determinaron los casos de uso de este y finalmente se obtuvieron 4 casos de uso, estos son: Modulo de ingreso, módulo de diagnosticar pega y módulo de ejecución de la solución específica.

4.5 Fase 5: Diseño

En esta fase, se realizó la transformación de un modelo a otro; es decir, iniciando con cuatro modelos de tareas, cada uno de estos se transformó a un modelo de interfaz abstracta, a partir de este, se realizó una transformación y se obtuvo los modelos de interfaz concreta en donde se pudo definir los objetos necesarios para la construcción de un archivo XML para finalmente obtener el diseño del modelo de interfaz final con objetos dependientes de la plataforma Unity. Cabe resaltar que en este modelo final el diseñador puede hacer las modificaciones que considere necesarias y apropiadas para su diseño. Para realizar la transformación de un modelo a otro, fue necesaria la creación de reglas específicas; dichas reglas fueron evaluadas y puestas a prueba por el Doctor Gabriel Pedraza dando como resultado la transformación esperada.

4.6 Fase 6: Codificación e Integración

En esta etapa se definieron las tareas de implementación para el desarrollo de nuestro proyecto, teniendo en cuenta los datos obtenidos en la fase de diseño, las especificaciones en los requisitos funcionales y no funcionales y el conjunto de casos de uso definidos.

En este proyecto se realiza la codificación en lenguaje C# para la definición de las acciones del sistema dentro del entorno Unity, también se construye e integra archivos XML y Json para el acceso, modificación y actualización de datos.

4.7 Fase 7: Pruebas

En esta última fase, se ejecutó un plan de pruebas manuales, unitarias, de integración y de usuarios, mediante las cuales se midió la eficiencia del enfoque MBUID en el desarrollo de nuestro proyecto. Para esto se ponen a prueba las reglas de concretización propuestas para la transformación de cada modelo, así como también, se analiza la posibilidad de crear diferentes diseños de UI en el entorno Unity a partir de un archivo XML y por último evidenciar con usuarios el impacto de la interfaz tutor integrada al simulador.

5. Desarrollo del proyecto

Figura 4. Mapa Explicativo del Desarrollo del Proyecto

Para tener una visión general de la estructura que se le dio a este proyecto, se puede observar en el mapa explicativo de la figura 4, las 7 fases de la metodología se reunieron en 3 grandes pasos donde 1) Comprende la investigación y recolección de información para la construcción de la lista de toma de actividades para la solución del problema operacional pega por empaquetamiento como también el planteamiento de requisitos funcionales y no funcionales para la interfaz de usuario de entrenamiento tutor. Como resultado en este paso se da la construcción de modelos de tareas a través de Concur Task Tree (CTT) para establecer el proceso necesario para la resolución de un problema operacional. 2) Comprende los pasos para la concretización del modelo obtenido en 1); iniciando la concretización del modelo de tareas CTT a través de reglas propuestas que actuarán como caja negra donde la salida será un modelo de interfaz de usuario abstracto, de la misma forma la concretización de modelo de interfaz de usuario abstracto al modelo de interfaz de usuario concreto y como resultado de este paso, obtener un modelo de interfaz final, donde se establecen reglas propias y dependientes de la plataforma a aplicar el diseño a través de un archivo XML. 3) Comprende la construcción de la interfaz dentro de la plataforma dependiente Unity, la construcción e integración de un archivo Json que recolecta los pasos en concreto obtenidos en 1) para la resolución de una pega, y la integración del archivo XML obtenido en 2). El resultado del paso 3 es la obtención de una interfaz de usuario tutor integrada a un simulador para la perforación de pozos petrolíferos que permitirá la resolución de un problema en el proceso de perforación a través de un despliegue estructurado de información procedimental, en donde finalmente el operador podrá ajustar tanto el diseño como el contenido.

5.1 Investigación y documentación de la problemática a solucionar

Como primera medida, se realizó una investigación acerca de los diferentes problemas operacionales que se presentan a la hora de realizar actividades de perforación petrolera, mediante esta fue posible definir el problema operacional especifico a tratar. Como resultado de esta investigación se decidió dar solución al problema operacional **pega por empaquetamiento** ya que es el problema más presentado en el proceso de perforación y a su vez el que más costos genera; por esta razón, se considera necesario y de gran utilidad para la industria petrolera que los perforadores tengan una buena preparación al respecto.

Como se mencionó en la introducción de este libro; es de gran importancia brindar al aprendiz las tareas necesarias para llevar a cabo un objetivo, por esta razón se procedió a investigar acerca de lo que es información procedimental y los diferentes tipos de información procedimental que existen; definiendo, que el tipo de información procedimental necesaria para llevar a cabo nuestro proyecto era la información procedimental algorítmica.

Mediante ese tipo de información es posible brindar tanto el orden, como las condiciones que deben cumplirse con el fin de dar una solución apropiada al problema operacional, además mediante el uso reiterado por parte del aprendiz se facilita el dominio de la técnica y se desarrolla la capacidad de toma de decisiones.

Luego se define el diseño del modelo instruccional en donde vendrán definidos el procedimiento y despliegue de la información procedimental a través de una teoría cognitivista la cual es eficaz para ayudar a los estudiantes a dominar el contenido dentro del proceso de perforación, así como cuando el aprendiz no trae ningún conocimiento previo o básico para la resolución de un problema.

La estructura propuesta parte de la premisa de la organización jerárquica desde lo general a lo particular, donde permitirá al aprendiz ejecutar soluciones básicas y que se ejecutan en el campo de la perforación para cualquier tipo de problema operacional, hasta evaluar particularmente el problema en cuestión.

Finalmente se procedió a buscar la información procedimental necesaria para dar una solución óptima al problema operacional pega por empaquetamiento y poder definir las actividades de toma de decisiones apropiadas, mediante las cuales el perforador podrá dar solución al problema operacional presentado. Las actividades de toma de decisiones establecidas y su estructura para dar solución al problema operacional presentado se encuentran en el apéndice A.

5.2 Capturar los requisitos de usuario

Como primera medida se realiza el levantamiento de requisitos identificando las necesidades del negocio, así como la solución a las mismas, con el fin de cumplir el alcance propuesto para este proyecto; cabe recordar, que este proyecto parte de un proyecto preliminar realizado por estudiantes de la escuela de Ingeniería de Petróleos. La especificación de requisitos se llevó a cabo con la colaboración del profesor Gabriel Pedraza Ferreira PhD en Informática y el profesor Luis Eduardo Bautista Rojas Magister en Ingeniería de Sistemas en Informática. A continuación, se muestra el documento de requisitos funcionales y no funcionales basado en la norma IEEE 830-1998 - IEEE Recommended Practice for Software Requirements Specifications.

5.2.1 Requerimientos funcionales

Número	**RF-01**
Nombre	Ingresar al tutor.
Descripción	El usuario u operador podrá ingresar a la interfaz de usuario tutor.
Restricciones	El usuario deberá estar ubicado en la cabina ejecutando el proceso de perforación.
Prioridad	Alta

Número	**RF-02**
Nombre	Cerrar Tutor.
Descripción	El usuario u operador podrá cerrar la interfaz de usuario Tutor.
Restricciones	Debe estar abierta la interfaz gráfica del Tutor.
Prioridad	Alta

Número	**RF-03**
Nombre	Evaluar estado del proceso de perforación.
Descripción	• *Si es un estado positivo:* indica que no se presenta ningún problema en el proceso de perforación. • *Si es un estado negativo*: indica que el usuario u operador observa algún tipo de anomalía en el proceso de perforación.
Restricciones	Debe estar abierta la interfaz de usuario del Tutor.
Prioridad	Alta

Número	**RF-04**
Nombre	Mostrar guía para solución de problemas
Descripción	El usuario podrá seguir una serie de pasos que proporciona la interfaz de usuario Tutor para resolver el problema que se esté presentando en el proceso de perforación.
Restricciones	Debe estar abierta la interfaz de usuario del Tutor.
Prioridad	Alta

Número	**RF-05**
Nombre	Ejecutar diagnóstico del problema.
Descripción	El usuario podrá enviar el estado de su proceso de perforación antes y después de la pega al sistema, para que pueda conocer el posible problema que tenga.
Restricciones	Debe estar abierta la interfaz de usuario del Tutor.
Prioridad	Alta

5.2.2 Requerimientos no funcionales.

Numero	**RNF_01**
Nombre	Niveles de solución de problemas.
Descripción	La interfaz de usuario Tutor deberá estar construida mediante una serie de niveles de solución del problema que se esté presentando. • *Nivel de ingreso a la interfaz:* el usuario conocerá la interfaz de usuario. • *Nivel de solución básica:* si el usuario indica que tiene un problema de pega, el sistema proporcionará una solución básica para cualquier tipo de pega.

Descripción	• *Nivel de diagnóstico de pega:* si el usuario indica que aún no se ha resuelto su problema, el sistema enviará un cuestionario para captar el estado de perforación antes y después del problema que se está presentado para que éste pueda arrojar un diagnóstico del posible tipo de pega. • *Nivel de Solución específica:* si el usuario ya conoce el tipo de pega que se está presentando, el sistema proporcionará la solución específica que se ejecuta en el campo de perforación para solucionar dicho problema.
Restricciones	
Prioridad	Alta

Numero	**RNF-02**
Nombre	Modificación de datos
Descripción	El conjunto de datos que arroja la interfaz de usuario y que comprende el diseño y el contenido, podrán ser modificados por cualquier tipo de usuario.
Restricciones	
Prioridad	Alta

Numero	**RNF-03**
Nombre	Actualización de datos
Descripción	El conjunto de datos de arroja la interfaz de usuario y que comprende el diseño y el contenido y que hayan sido modificados (ver RNF-02.) se actualizarán al sistema de forma inmediata.
Restricciones	Haber modificado los datos (ver RNF-02).
Prioridad	Alta

Numero	RNF-04
Nombre	Tiempo de respuesta de sistema en objetos de control.
Descripción	El tiempo de respuesta cuando un usuario ejecute una acción a través de un objeto de control o cuadro de comando, deberá ser máximo de 0.1 segundos.
Restricciones	
Prioridad	Alta

5.3 Definición de contexto PIM

Definición de actor: Un actor es quien adopta el rol de uso del sistema e interactúa con él, puede comprender desde una persona hasta otros dispositivos.

5.3.1 Actor aprendiz u operador. Se define al actor aprendiz como aquel que hará uso de la interfaz de usuario Tutor, que podrá hacer uso de distintas funciones como:

- Hacer uso de la interfaz de usuario tutor una vez haya ingresado a la cabina de perforación y tenga algún problema en el proceso de perforación.
- Evaluar el estado del proceso de perforación.

5.3.2 Actor administrador. Se define al actor administrador como aquel que plantea el escenario que el aprendiz va a resolver como también quien puede editar el contenido para la resolución de un problema en el proceso de perforación.

5.3.3 Actor sistema. El actor sistema representa la interfaz de usuario de entrenamiento como tutor y las acciones que este permitirá desarrollar.

5.4 Especificación de los requisitos PIM

Los diagramas de caso de uso representan un esquema en la relación entre un actor o actores y un sistema. Los casos de uso permiten reunir los requisitos funcionales del sistema y observar en detalle las acciones de los actores al interactuar con este. A continuación, se presentan los distintos casos de uso que se encuentran en el desarrollo de la interfaz de usuario Tutor.

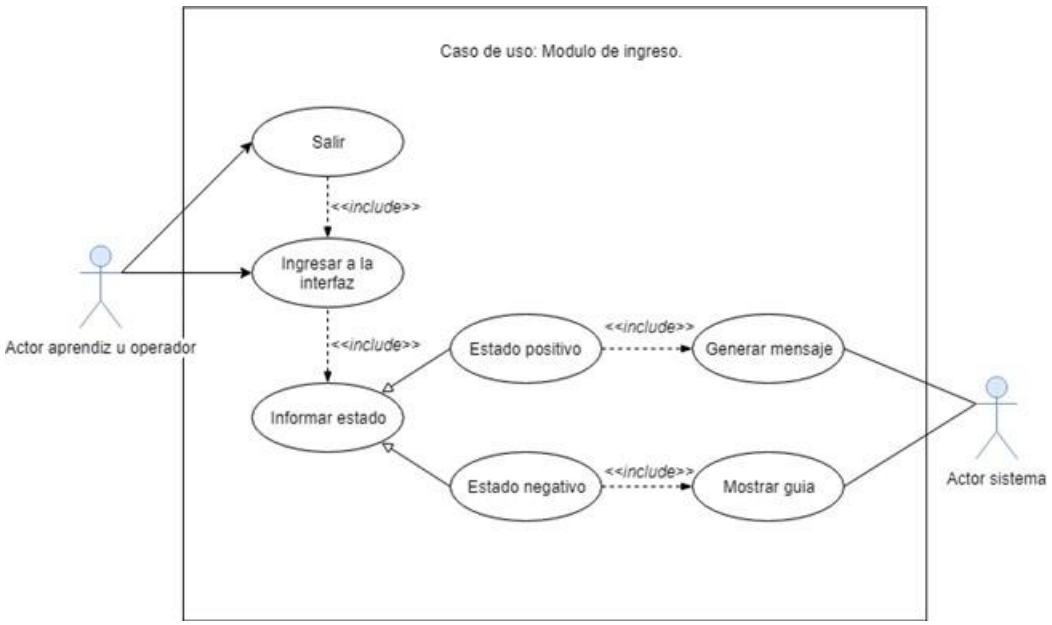


Figura 5. Caso de uso Modulo de Ingreso

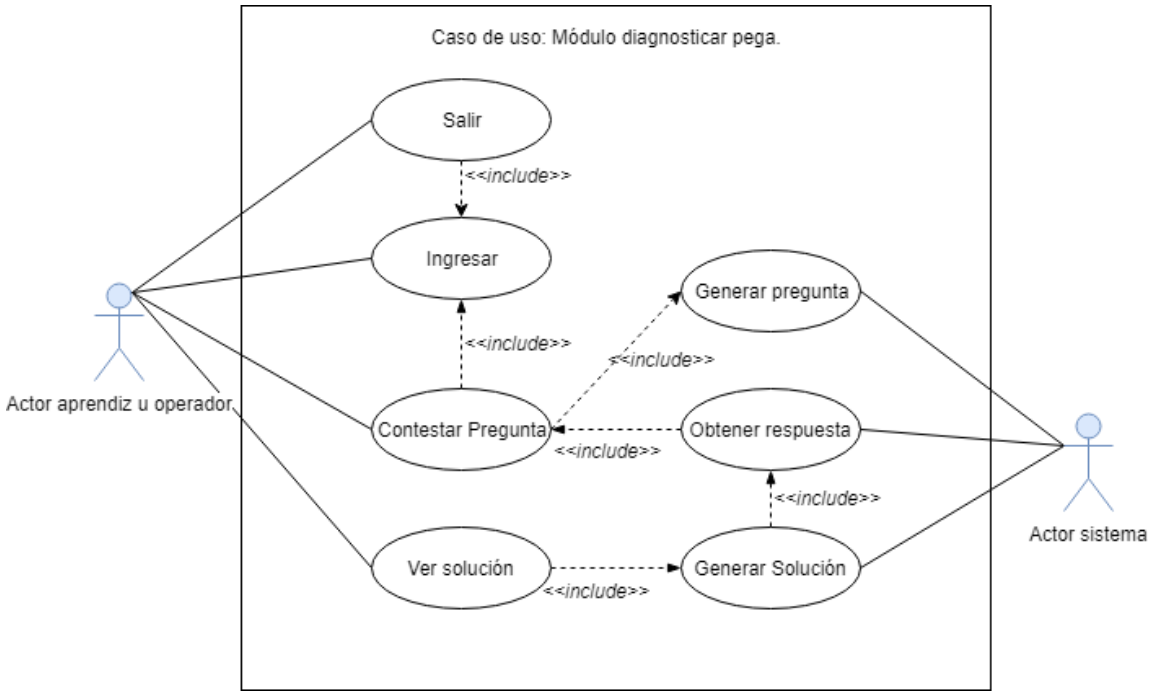


Figura 6. Caso de uso Modulo Diagnosticar Pega.

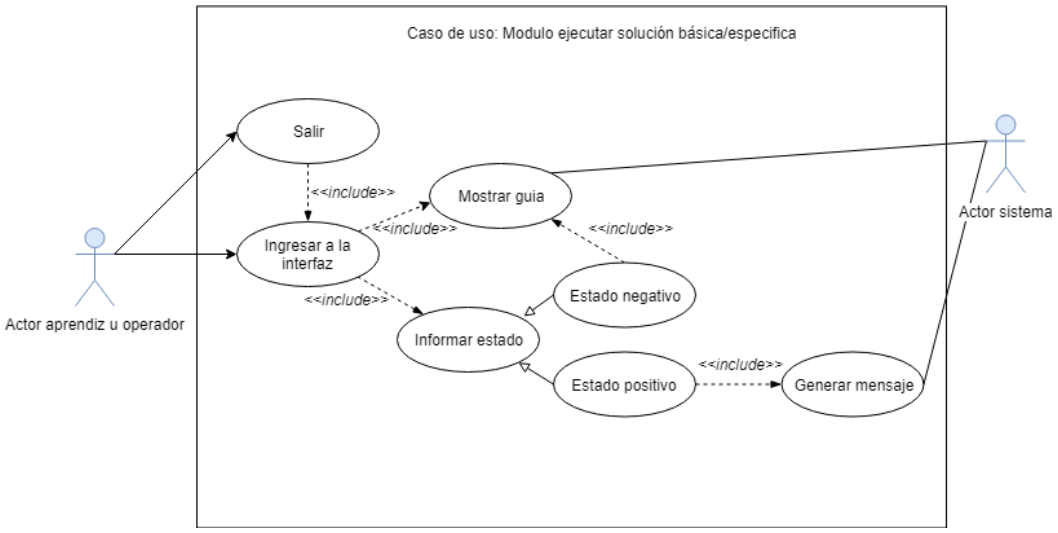


Figura 7. Caso de uso Modulo Ejecutar Solución Básica y Especifica

5.5 Diseño

Como se ha mencionado anteriormente el desarrollo de interfaz de usuario basada en modelos (MBUID) requiere una serie de modelos los cuales permiten identificar y definir el sistema claramente. Estos modelos inician en una fase de alta abstracción y son transformados en modelos más concretos, hasta obtener el código de la interfaz de usuario final.

El paso de un modelo a otro requiere Reglas de concretización, para el desarrollo de este proyecto fue necesaria la creación de reglas propias.

5.5.1 Modelo de tareas. Como primera medida, se crea el modelo de tareas, en el desarrollo de este proyecto se usó la herramienta Concur Task Tree (CTT). Mediante este modelo, se representan las tareas que el usuario final debe realizar en la interfaz de usuario Tutor para dar solución al problema operacional “Pega de tubería por empaquetamiento” de manera efectiva, en el desarrollo de este modelo se presentaron cuatro modelos de tareas (Ingresar_Interfaz_Tutor, Ejecutar_Solución_Basica, Diagnosticar_Pega, Ejecutar_Solución_Especifica); a continuación, se enseña de forma detallada la construcción el modelo de tareas en la Figura 8 para el Ingreso a la interfaz de usuario. Los demás modelos de tareas se pueden consultar en el Apéndice A.

5.5.1.1 Modelo de tareas de ingreso a la interfaz de usuario Tutor.

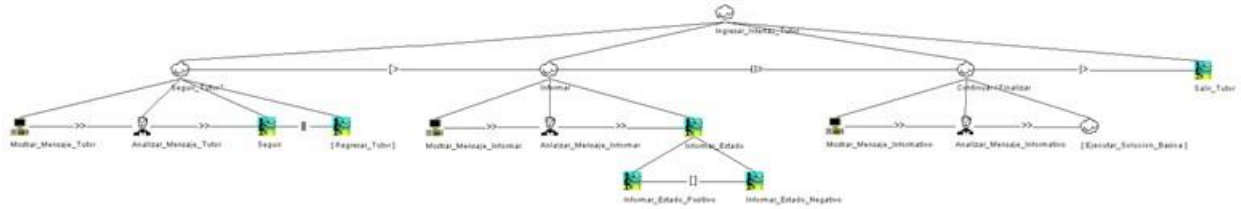


Figura 8. Modelo de Tareas para el Ingreso a la Interfaz de Usuario Tutor

Tarea Abstracta Ingresar_Tutor: Mediante esta tarea se da el ingreso a la interfaz de usuario Tutor por parte del usuario final para iniciar la solución del problema operacional presentado.

- ***Subtarea Abstracta Seguir_Tutor*:*** Esta tarea es una tarea iterativa (se repetirá hasta que finalice por completo), se encarga de brindar información al usuario y permitir que este continúe en el proceso de solución al problema operacional presentado.
    - ***Subtarea de Sistema Mostrar_Mensaje_Tutor:*** Esta es una tarea de tipo "Feedback"; mediante esta, se muestra información al usuario; en este caso, la interfaz de usuario Tutor da la bienvenida al usuario.
    - ***Subtarea de Usuario Analizar_Mensaje_Tutor:*** Esta es una tarea de tipo "Planning"; mediante esta, el usuario analiza la información brindada por la interfaz de usuario Tutor.
    - ***Subtarea de Interacción Seguir:*** Esta es una tarea de tipo "Control"; mediante esta, el usuario solicita continuar con el proceso para dar solución al problema operacional que se está presentando.

- ***Subtarea de Interacción [Regresar_Tutor]:*** Esta es una tarea de tipo "Control"; mediante esta, el usuario solicita regresar en el proceso y verificar la información brindada por la interfaz de usuario Tutor, si lo desea.

- ***Subtarea Abstracta Informar:*** Esta tarea se encarga de obtener el estado de la perforación para posteriormente enviar la información obtenida a la siguiente tarea y así continuar con el proceso de solución al problema operacional presentado.

  - ***Subtarea de Sistema Mostrar_Mensaje_Informar:*** Esta es una tarea de tipo "Feedback"; mediante esta, la interfaz de usuario Tutor solicita al usuario informar el estado actual de la perforación.

  - ***Subtarea de Usuario Analizar_Mensaje_Informar:*** Esta es una tarea de tipo "Planning"; mediante esta, el usuario analiza la información brindada por la interfaz de usuario Tutor.

  - ***Subtarea de Interacción Informar_Estado:*** Mediante esta tarea el usuario selecciona el estado actual de la perforación, este puede ser Positivo o Negativo.

  - ***Subtarea de Interacción Informar_Estado_Positivo:*** Esta es una tarea de tipo "Control"; mediante esta, el usuario selecciona que el estado actual de la perforación es Positivo; es decir, la perforación avanza con normalidad.

  - ***Subtarea de Interacción Informar_Estado_Negativo:*** Esta es una tarea de tipo "Control"; mediante esta, el usuario selecciona que el estado actual de la perforación es negativo; es decir, la perforación no avanza o avanza con anormalidad (muy lento, alarmas, fugas, etc.).

- ***Subtarea Abstracta Continuar/Finalizar:*** Mediante esta tarea el usuario puede continuar o finalizar el proceso de solución al problema operacional presentado, esto depende de la información obtenida de la tarea inmediatamente anterior.
- ***Subtarea de Sistema Mostrar_Mensaje_Informativo:*** Esta es una tarea de tipo "Feedback". Esta tarea tiene dos opciones:

Si el *Estado actual del problema operacional es Positivo;* la interfaz de usuario Tutor, informa al usuario que puede continuar su perforación con normalidad.

Si el *Estado actual del problema operacional es Negativo;* la interfaz de usuario Tutor, informa al usuario que a continuación se darán unos pasos básicos para dar solución al problema operacional presentado.

- ***Subtarea de Usuario Analizar_Mensaje_Informativo:*** Esta es una tarea de tipo "Planning"; mediante esta, el usuario analiza la información brindada por la interfaz de usuario Tutor.
- ***Subtarea Abstracta [Ejecutar_Solución_Basica]:*** Esta es una tarea de tipo "SearchInformation", esta es una tarea opcional; es decir, se realizará según la información obtenida anteriormente.

En caso de ser realizada, mediante esta tarea el usuario solicita continuar para dar una solución básica al problema operacional presentado.

- ***Subtarea de Interacción Salir_Tutor:*** Esta es una tarea de tipo "Control"; mediante esta, el usuario solicita salir de la interfaz Tutor en cualquier momento del proceso de solución al problema operacional.

5.5.2 Modelo de Interfaz de Usuario Abstracta (AUI). En esta fase del diseño de interfaz de usuario basado en modelos se describe la interfaz de usuario Tutor usando un alto nivel de abstracción; además, mostrara los contenedores y componentes abstractos obtenidos, y estos últimos con sus correspondientes facetas.

5.5.2.1 Reglas de concretización de Modelo de Tareas a Modelo de Interfaz de Usuario Abstracta: Para la realización de la transformación del modelo de tareas al modelo de interfaz abstracta fue necesaria la creación de las siguientes reglas; mediante estas reglas, se definen los contenedores y componentes abstractos que agruparan la ejecución de las tareas definidas en el modelo anterior. A continuación, un ejemplo en la Figura 9 dónde se puede apreciar la transformación del Modelo de Tareas al Modelo de Interfaz Abstracta aplicando las Reglas de concretización.

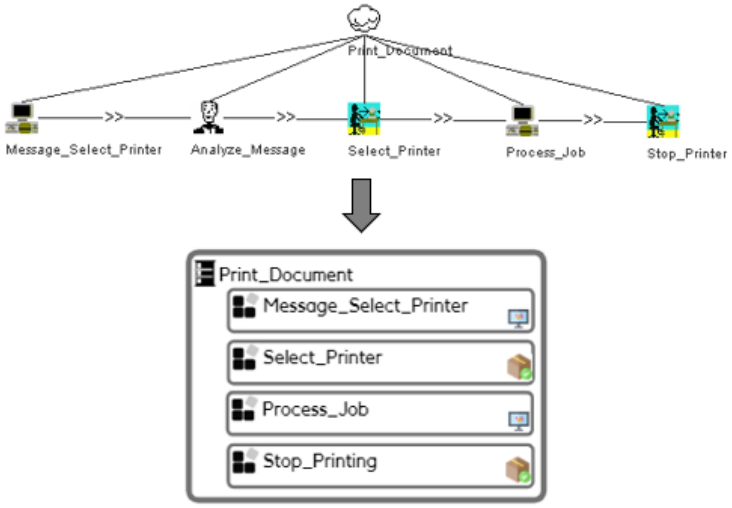


Figura 9. Ejemplo de Modelo de Tareas Transformado a Modelo de Interfaz Abstracta.

5.5.2.1.1 Regla de división: Esta transformación admitirá la segmentación de la presentación total (tarea de alto nivel), en un conjunto de interfaces individuales que conformarán las tareas de bajo nivel permitiendo diferencias de tamaño y resoluciones en la pantalla de destino final.

5.5.2.1.2 Regla de Orden: La transformación del modelo de tareas al modelo abstracto se realizará secuencialmente, garantizando el acatamiento de los operadores temporal de la notación CTT.

Verticalmente se iniciará con tareas de alto nivel hasta las subtareas y horizontalmente se transformará de izquierda a derecha.

5.5.2.1.3 Regla de identificación: A continuación, se define el icono mediante el cual se identificarán tanto el contenedor componentes abstractos, así como cada una de las facetas de este último.

-  Icono mediante el cual se identificarán los Contenedor Abstracto (AC).
-  Icono mediante el cual se identificarán los Componente de Interacción Abstracto (AIC).
-  Icono mediante el cual se identificará la Faceta de Control.
-  Icono mediante el cual se identificará la Faceta de Salida.
-  Icono mediante el cual se identificará la Faceta de Entrada.
-  Icono mediante el cual se identificará la Faceta de Navegación.

- Toda tarea de interacción con la propiedad asociada “SingleSelection” transformada, se representará mediante un contenedor punteado **(- - - -).**
- Al realizar la transformación los AIO´s se definirán mediante el nombre de la respectiva tarea transformada, este nombre debe ser idéntico al usado en el modelo de tareas; es decir, si en dicho nombre se encuentra también la relación temporal, como es el caso de las relaciones temporales iterativas ***T1**** y opcionales ***[T1]*** siendo T1 el nombre de la tarea; esta relación, también debe ser parte del identificador de los AIO´s.

5.5.2.1.4 Regla de Posición.

- El Contenedor Abstracto (AC) contiene tanto Contenedores Abstractos (AC) como Componentes de Interacción Abstracto (AIC), los cuales se ubicarán de manera descendente y su forma será rectangular.
- Cuando se realice la transformación el icono de los contenedores y componentes abstractos, se ubicará en la parte superior izquierda del respectivo contenedor.
- Cuando se realice la transformación el icono de las facetas del Componente de Interacción Abstracto (AIC) se ubicará en la parte inferior derecha del respectivo contenedor.
- El Identificador de cada AIO se ubicará junto al icono del contenedor y componente abstracto.

5.5.2.1.5 Regla de concretización.

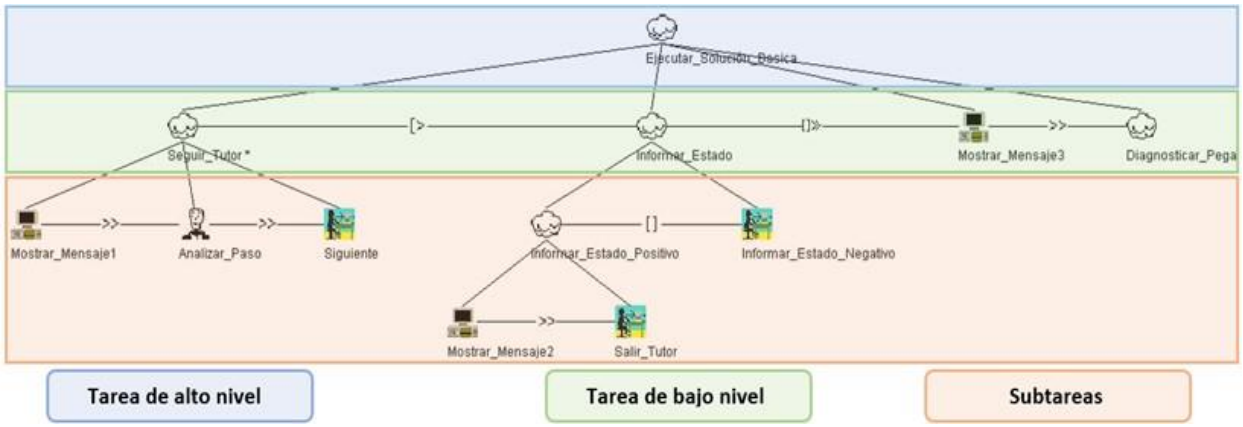


Figura 10. Identificación Según Nivel de Tareas.

Tomando como guía visual la Figura 10, se establecen las siguientes reglas:

- **Tarea de alto nivel:** La tarea de alto nivel se convierte en un contenedor AC, siendo este el contenedor de primer orden de tareas de bajo nivel y subtareas.
- **Tarea de bajo nivel:** Si una tarea de bajo nivel se descompone en subtareas, esta se transforma en un Contenedor Abstracto (AC) siendo este el contenedor de segundo orden; de lo contrario se convertirá en un Componente de Interacción Abstracto (AIC).
- **Subtareas**: serán contenidas por el respectivo contenedor abstracto AC que se generó al realizar la transformación de la tarea de bajo nivel. Si la subtarea se descompone en subtareas, está se transformará en un Contenedor Abstracto (AC) siendo este un contenedor de tercer orden; si la subtarea no se descompone en subtareas, esta se transformará en Componente de Interacción Abstracto (AIC), identificando la faceta a la cual pertenece cada subtarea. Para ello se debe tener presente el tipo de tarea en notación CTT de la siguiente forma:

- Las tareas de interacción con el tipo de tarea asociado “Control” y “SingleSelection” se convertirán en Componentes de Interacción Abstractos (AIC) con faceta de control.
- Las tareas de interacción con el tipo de tarea asociado “Edit” se convertirán en Componentes de Interacción Abstractos (AIC) con faceta de entrada.
- Las tareas de sistema con el tipo de tarea asociado “Feedback” se convertirán en Componentes de Interacción Abstractos (AIC) con faceta de salida.
- Las tareas abstractas con el tipo de tarea asociado “SearchInformation” se convertirán en Componentes de Interacción Abstractos (AIC) con faceta de navegación, permitiendo acceder al siguiente CTT
- Las tareas de interacción con el tipo de tarea asociado “DetailOnDemand” serán eliminadas en la transformación ya que éstas no generan valor agregado a la interfaz abstracta.
- Las tareas de usuario serán eliminadas en la transformación ya que éstas no generan valor agregado a la interfaz abstracta.

A continuación, se enseña el modelo de interfaz de usuario abstracto para el ingreso a la interfaz de usuario Tutor, obtenido al hacer uso de las reglas de concretización expuestas anteriormente. Los demás modelos AUI se pueden ver en el Apéndice B.

5.5.2.2 Modelo Interfaz de Usuario Abstracta para el ingreso a la interfaz Tutor

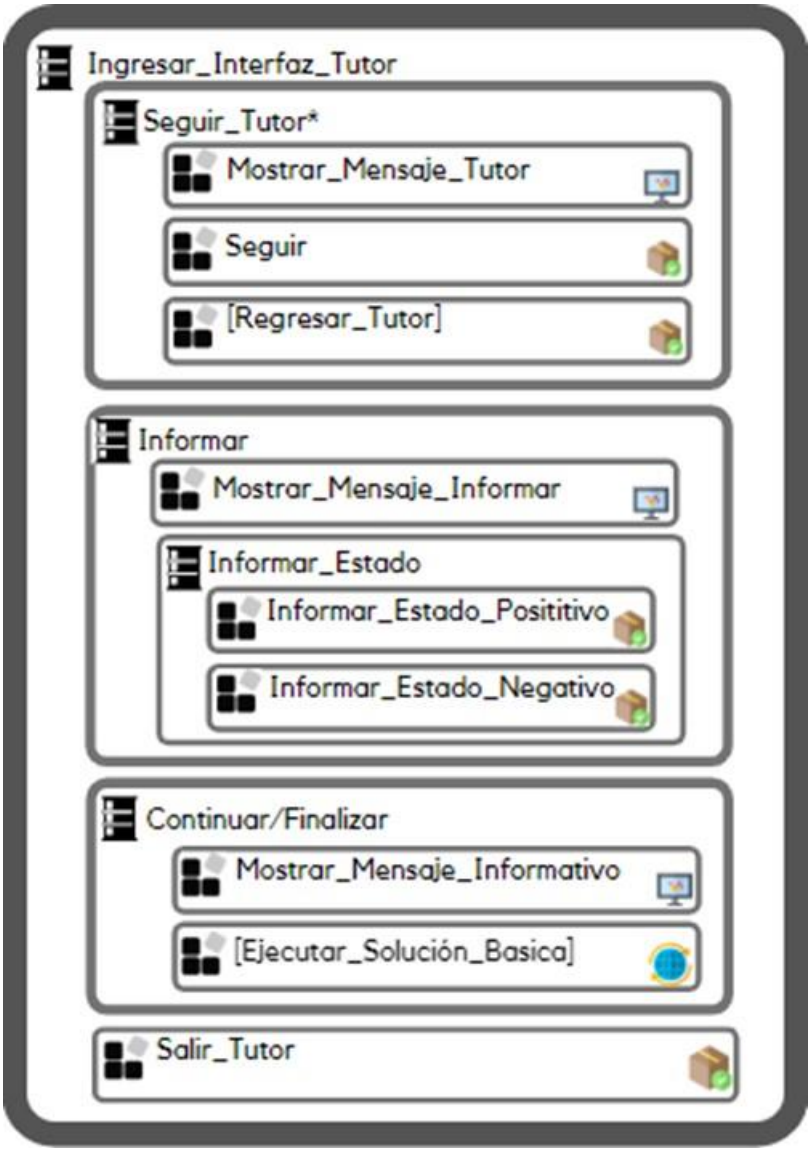


Figura 11. Modelo AUI para Ingresar a la Interfaz de Usuario Tutor.

5.5.3 Modelo de Interfaz de Usuario Concreta (CUI). Al realizar la transformación de un modelo de interfaz abstracto a uno concreto se materializa la definición abstracta anterior en cada contexto de uso, mediante widgets y navegación entre interfaces tal como se muestra en el ejemplo de la Figura 12; a continuación, se definen las reglas a aplicar para realizar la transformación.

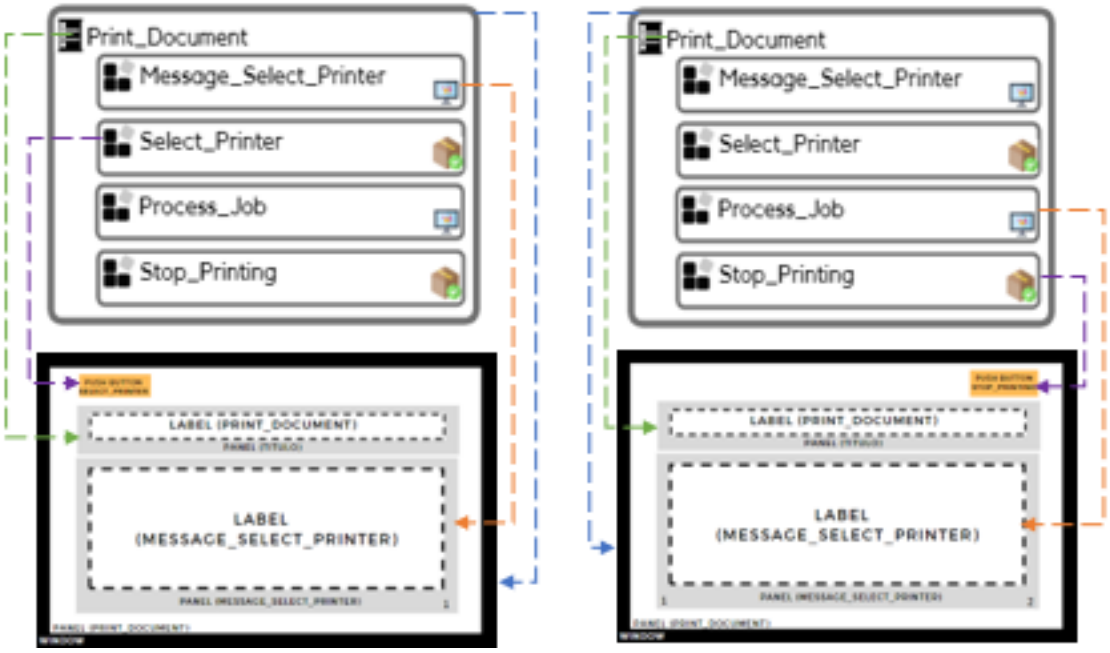


Figura 12. Ejemplo, Transformación de AUI a CUI.

5.5.3.1 Reglas de concretización de Modelo de interfaz abstracto a modelo de interfaz concreto: Para realizar esta transformación se crearon y aplicaron tres tipos de reglas, reglas de selección, reglas de enlace y reglas de identificación; es necesario aclarar que la posición de los objetos la define el diseñador de la interfaz.

5.5.3.1.1 Reglas de selección.

- El Contenedor Abstracto (AC) de primer grado será un objeto de diálogo “Window” dentro del cual se creará un objeto tipo “Panel”, este contendrá todos los elementos concretos asignados.
- El identificador del contenedor de primer grado AC será un objeto tipo “Label” que se mantendrá a lo largo de la transformación de dicho contenedor.
- Todo Componente de Interacción Abstracto AIC con faceta de salida será un objeto tipo “Label”, cada objeto tipo “Label” está contenido por un objeto tipo “Panel” diferente.
- Todo Componente de Interacción Abstracto (AIC) con faceta de control o faceta de navegación, se asignará cómo un objeto tipo “Push Button”, este puede crearse como una imagen o como un botón genérico (botón de comando el cual contiene un “Label”, también llamada “Etiqueta” o “Caption").
- Todo Componente de Interacción Abstracto AIC con faceta de entrada será un objeto tipo “Input”, también llamado “Campo o Cuadro de Texto”.
- Todo Componente de Interacción Abstracto (AIC) en faceta de control y alojado en un contenedor punteado, será un objeto tipo “Group Box”

5.5.3.1.2 Regla de enlace.

- Los Contenedores Abstractos (AC) conformados por un AIC en faceta de salida, seguido de un AIC en faceta de control se visualizarán en un mismo display dentro del objeto de dialogo “window”; esta regla también aplica si se tiene Contenedores Abstractos (AC) conformados por un AIC en faceta de salida seguido de un AC conformado por AIC’s en faceta de control.

- Todo objeto “Push Button” transformado definirá un cambio de display dentro del objeto de dialogo “window”.
- Los AIO´s en los que su ID tiene asociada la relación temporal [T1], podrán visualizarse o no, por lo tanto, es necesario realizar la transformación para las dos opciones.
- Todo AIC de segundo orden en faceta de control se mantendrá a lo largo de la transformación y podrá ejecutarse en cualquier momento.

5.5.3.1.3 Regla de identificación.

- Al realizar la transformación de cualquier Objeto de Interacción Abstracto (AIO) en los que su ID tiene asociada la relación temporal **T***; se indicará mediante (*) que el contenido del display es dinámico.

A continuación, se enseña el modelo de interfaz de usuario concreto para el ingreso a la interfaz de usuario Tutor, obtenido al hacer uso de las reglas de concretización expuestas anteriormente. Los demás modelos AUI se pueden ver en el Apéndice C.

5.5.3.2 Modelo de Interfaz de Usuario Concreta para ingresar a la Interfaz Tutor: Este modelo, enseña los diferentes objetos que se encuentran en cada display, el número ubicado en los extremos inferiores de cada display no hace parte de la transformación, este se usa para ayudar a la comprensión en el cambio de un display a otro donde n indica el número de veces que puede cambiar el contenido del display al este ser dinámico; cabe recordar, que el identificador (*) indica

que dicho display es iterativo; es decir, su contenido es dinámico, tal como se muestra en la figura 13.

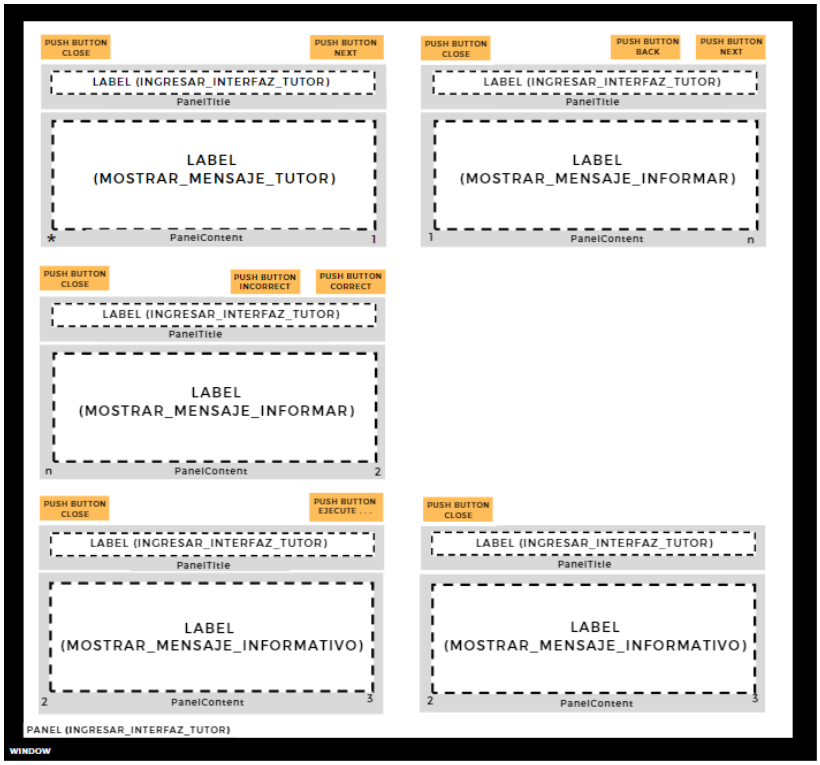


Figura 13. Modelo CUI para Ingresar a la Interfaz Tutor

A continuación, se especifica el nombre de cada display.

- **Seguir Tutor**: Conformado por los displays identificados como (*-1) y (1 - n).
- **Informar:** Conformado por los displays identificado como (n - 2).
- **Continuar/Finalizar:** Conformado por los displays identificado como (2 - 3).

5.5.4 Modelo de interfaz concreto a modelo de interfaz final. Habiendo terminado la transformación al modelo de interfaz concreto CUI, es necesario identificar cada uno de los objetos y enumerar la cantidad de objetos diferentes que se obtuvieron; esto para evitar redundancias y saber cuáles y cuantos objetos de interfaz de usuario se van a crear en el entorno Unity. Posteriormente se hace una comparación de los objetos concretos genéricos obtenidos anteriormente respecto a los objetos y componentes que proporciona el entorno de Unity, para la construcción del diseño. Finalmente se integra el valor de cada parámetro del diseño de los objetos de interfaz de usuario por medio de un archivo XML. Por ejemplo, para el modelo CUI ingresar a la interfaz tutor (ver Figura 13), se obtuvo: un objeto tipo Window, un objeto tipo panel (Ingresar_Interfaz_Tutor) y un conjunto de displays especificados a continuación.

Tabla 6. *Objetos por Cada Display de CUI Ingresar a la Interfaz Tutor.*

Display	**Nombre**	**Objetos CUI**
Display 1	Seguir_Tutor	PanelTitle PanelContent Label (Ingresar_Interfaz_Tutor). Label (Mostrar_Mensaje_Tutor). Botón Close. Botón Next. Botón Back.

Tabla 6. *Continuación*

Display	Nombre	Objetos CUI
Display 2	Informe	PanelTitle PanelContent Label (Ingresar_Interfaz_Tutor). Label (Mostrar_Mensaje_ Informar). Botón Close. Botón Correct. Botón Incorrect.
Display 3	Continuar/Finalizar	PanelTitle PanelContent Label (Ingresar_Interfaz_Tutor). Label (Mostrar_Mensaje_ Informativo). Botón Close. Botón Ejecute.

Por lo que eliminando redundancias en la creación de objetos de interfaz de usuario y ordenándolos jerárquicamente para poder crearlos en el entorno Unity se obtiene:

Tabla 7. *Objetos de CUI Ingresar a la Interfaz Tutor, Sin Redundancias.*

Lista de objetos del modelo CUI para ingresar a la Interfaz de Usuario Tutor
1 Window
1 Panel (Ingresar_Interfaz_Tutor)
1 Panel (Titulo)
Label (Ingresar_Interfaz_Tutor)
1 Panel (Mostrar_Mensaje_Tutor))

Tabla 7. *Continuación*

Lista de objetos del modelo CUI para ingresar a la Interfaz de Usuario Tutor
Label (Mostrar_Mensaje_Tutor)
1 botón (Seguir)
1 botón (Atras)
1 botón (Negativo)
1 botón (Positivo)
1 botón (Ejecutar)
1 botón (Close)

A continuación, se identifican los objetos de interfaz de usuario concreta que son correspondientes a los objetos de interfaz de usuario en el entorno Unity y las distintas propiedades de cada uno:

Tabla 8. *Objetos de Modelo CUI vs Objetos en el Entorno Unity.*

Objeto CUI	**Objeto de CUI en el entorno Unity**	**Propiedades del objeto en el entorno Unity**
Window (Ventana)	Canvas	Renderización pre- definida: Screen Space-Camera (Espacio de la pantalla-Camara)
Panel	Panel	• Position: Left, Top, Right y Bottom. • Color: Normal. • Imagen: BackgroundImage.
Label	Label	Alignment, Font size, Color, Font style y Font.
Botón/Cuadro de comando/ *Push Button*	Button	• Position: Left, Top, Right y Bottom. • Color: Normal, Highlighted, Pressed y disabled. • Image: BackgroundImage.

Tabla 8. *Continuación*

Objeto CUI	Objeto de CUI en el entorno Unity	Propiedades del objeto en el entorno Unity
Botón/Cuadro de comando/ *Push Button*	Button	• Label: Alignment, Font size, Color, Font style y Font.
Texto de entrada	Input Text	• Position: Left, Top, Right y Bottom. • Color: Normal, Highlighted, Pressed y disabled. • Placeholder. • Label: Alignment, Font size, Color, Font style y Font.
GroupBox	Grid Layout Group	• Position: Left, Top, Right y Bottom. • Padding: PaddingTop, PaddingRight, PaddingLeft, PaddingBottom. • CellSize_x, CellSize_y. • Spacing_x, Spacing_y. • Color: Normal, Highlighted, Pressed y disabled. • Label: Alignment, Font size, Color, Font style y Font

Para extraer las características de cada uno de los objetos concretos que se usan, se tuvieron en cuenta aquellas con mayor importancia y que dieran un cambio significativo al diseño

Tamaño y posición: El tamaño de cualquier objeto de interfaz de usuario en el entorno Unity se puede ajustar mediante el componente *Rect Transform*, el cual permite ajustar el área de trabajo en cuanto a pivotes, anclaje (anchors), rotación, escala y posición en los objetos. Para este caso, la característica más importante es el anclaje, en la cual se tienen dos tipos de anclaje:

- Anclaje en un punto.

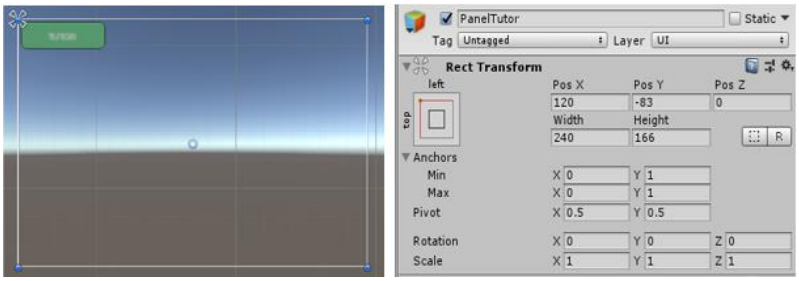


Figura 14. Anclaje en un Punto en el Entorno Unity.

Nota: Al lado derecho se observan los datos del objeto en el inspector de Unity y al lado izquierdo la posición de las anclas dentro de la escena.

Como se observa en la figura 14, este tipo de anclaje toma un punto de referencia y a partir de este se define la posición del objeto mediante cuatro propiedades importantes; estos son: *Pos X, Pos Y, Width y Height*, ubicados en el inspector del entorno Unity.

- Anclaje en rectángulo.

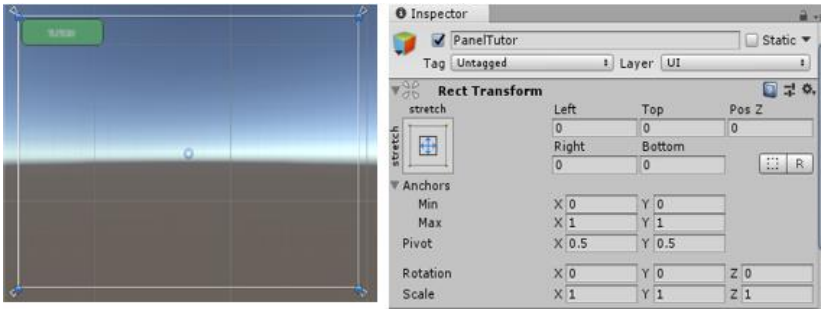


Figura 15. Anclaje en Rectángulo en el Entorno Unity.

Nota: Al lado derecho se observan los datos del objeto en el inspector de Unity y al lado izquierdo la posición de las anclas dentro de la escena.

Este tipo de anclaje toma la posición de su objeto padre inmediato (en el ejemplo de la figura 15, el padre inmediato del *panelTutor* es el Canvas). Para definir la posición del objeto, se tienen de nuevo cuatro propiedades importantes: Left, Top, Right y Bottom, los cuales indican la distancia del objeto que se está tratando al objeto padre.

Una vez analizados los tipos de anclaje, se decide adoptar por el anclaje en rectángulo, ya que permite ajustar los objetos en cuatro puntos (ver Figura 16), donde cada punto medirá la distancia al límite del objeto padre que lo contiene mediante el ajuste de las cuatro propiedades de posición mencionadas anteriormente y además evitar perderse dentro del escenario ya que en el caso del anclaje en un punto permite números negativos para ajustar las propiedades de los Ejes X y Y.

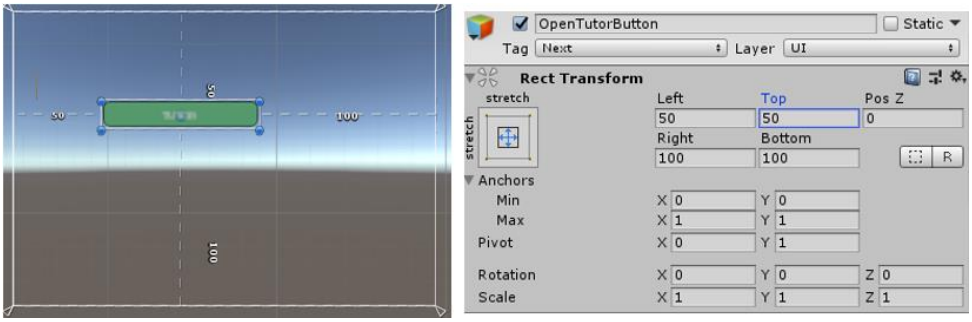


Figura 16. Definición de Left, Top, Right y Bottom para un Botón.

Para el caso de *Label* o etiqueta, la posición y el tamaño se puede ajustar mediante las propiedades dadas por este componente: *alingment* y *fontsize*. No es necesario reajustar el *Rect transform* ya que este tipo de objetos toma directamente la posición de su padre inmediato, por ejemplo, en el caso de "LabelTutor" de la Figura 17, el padre inmediato es *Button* este le hereda la posición en la escena, por lo que los límites de este Label serán los límites del Button; por ello, es necesario conocer el padre inmediato de un objeto para tener presente el tamaño que el hijo va a adoptar.

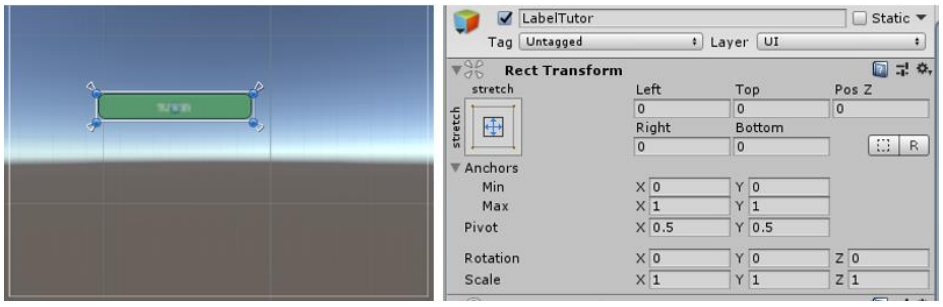


Figura 17. Label de un Botón: Hereda la Posición de su Padre Inmediato.

Color

Existen dos tipos de presentación de colores en los objetos, y depende del tipo de objeto. Si es un objeto estático (Panel o *Label*) requiere solo un color, que será el *Background* del objeto hijo en el caso del panel o el color del texto en el caso del *Label*. Si por el contrario es un objeto interactivo (Botón) es importante indicar el color de los cuatro estados de un bóton (*Normal Color, Highlighted Color, Pressed Color y Disabled Color*).

El color del objeto se puede obtener en cualquier formato convencional conocido (RGB, HSV o Hexadecimal). En este caso se decide usar el formato hexadecimal para asignar el color ya que el usuario solo necesitaría llenar una etiqueta del archivo XML (en el caso de los otros dos formatos son 3 campos; uno para cada color).

Ahora, si se desea aplicar opacidad al color del objeto, se debe hacer uso de la tabla de porcentaje de opacidades, esta tabla se puede encontrar en el Apéndice D.

Por ejemplo, cómo se observa en la Figura 18, si se quiere indicar el color rojo con hexadecimal #ff0000 con una opacidad del 60%, según la tabla de porcentaje de opacidades este porcentaje corresponde al prefijo de valor 99, concatenando los dos valores se obtiene: #ff000099.

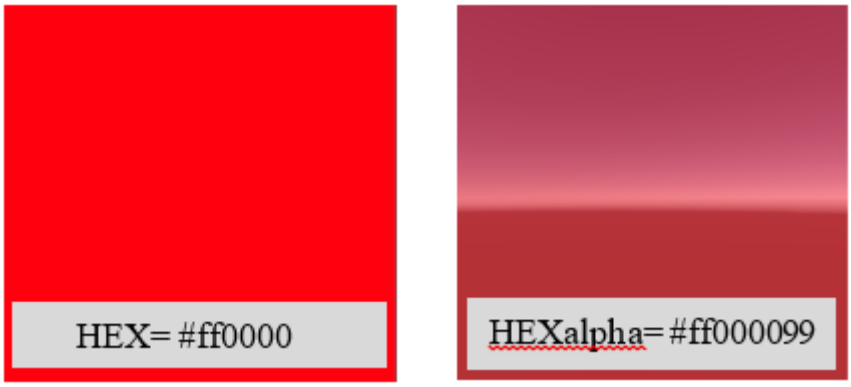


Figura 18. Concatenación del Color.

Imagen o Background: La imagen o background se puede asignar mediante una ruta de localización de la imagen, también se puede añadir color normal o en caso si es un objeto interactivo, los 4 colores ya mencionados.

5.5.4.1 Reglas de concretización del modelo CUI al archivo XML

5.5.4.1.1 Regla de orden: La transformación se hará de afuera hacia adentro; partiendo del panel más grande el cual reúne todos los elementos, luego se subdivide en displays y finalmente en los objetos que describirán las propiedades de estos.

5.5.4.1.2 Regla de selección.

- Todo objeto de diálogo "*Window*" será un objeto tipo Canvas en Unity, sin embargo, este no asignará especificaciones en el archivo de XML ya que la renderización de este (*Screen Space-Camera* (Espacio de la pantalla-Cámara)) es predefinida por lo tanto esta transformación se omite.
- El panel inmediatamente después del objeto de dialogo "*Window*", será considerado como el contenedor mayor, y será la etiqueta <Content>.
- Cada display creado dentro del panel mencionado en la regla anterior tendrá una etiqueta <Display> dentro de la etiqueta <Content>.
- Por cada grupo de objetos encontrado en cada display, se creará una etiqueta <Object>.
- En el caso de los *Label*, como siempre está contenido dentro de un objeto, no se considera como grupo de objetos; por lo tanto, no será una etiqueta tipo <Object> sino una propiedad dentro de <Object>.

- Por cada propiedad del objeto representado por la etiqueta <Object>, se creará dentro de ésta, una etiqueta con el nombre de dicha propiedad.

5.5.4.1.3 Regla de especificación de atributos para cada etiqueta: En esta regla se especifican los atributos que contiene cada etiqueta, mediante las cuales se describe el modelo final en XML.

Tabla 9. *Propiedades de las Etiquetas.*

Etiqueta	Descripción	Atributos	Especificaciones
<*Content*>	Etiqueta que representa el contenedor mayor, ésta etiqueta es única en cada XML.	ID Name	**ID**: funciona como un contador, según el orden de las transformaciones, debe empezar en cero. **Name**: corresponde al nombre que se le dio al modelo.
<Display>	Etiqueta que representa una ventana dentro del contenedor mayor, se crea uno por cada display del modelo de interfaz concreta.	Name ID	**ID**: es un contador que inicia en cero y corresponde al orden de las transformaciones de los displays que se especifican previamente. **Name**: corresponde al nombre exacto proporcionado a cada display.
<Object>	Etiqueta que representa al grupo de objetos que contiene el display, se crea una por cada objeto del modelo de interfaz concreto. Si el objeto se repite en el mismo display, se crea solo una etiqueta tipo *object.*	Type Name	**Type**: corresponde al grupo de objetos de Unity (Panel, Button, GroupBox o InputText). **Name**: corresponde el nombre exacto que se le dio en el modelo de interfaz concreto.

5.5.4.1.4 Regla de asignación del nombre de las propiedades y los posibles valores de Key para cada Objeto: Esta regla se realiza teniendo en cuenta la Tabla 9 mediante la cual se representan las etiquetas de los distintos tipos de objetos.

Tabla 10. *Propiedades de los Objetos de Interfaz de Usuario en el Entorno Unity.*

Propiedad	Posibles valores de Key	Tipos de objetos en los que pueden estar contenidos.
<Position>	Right, Left, Top, Bottom	Panel, Button, InputText y Grid Layout Group.
<Label>	Alignment, FontSize, Text, ColorLabel, FontStyle y Font	Panel, Button, InputText y Grid Layout Group.
<Color>	Normal	Panel, Button, InputText y Grid Layout Group.
	Highlighted, Pressed, Disabled	Button y Grid Layout Group.
<Placeholder>	Alignment, FontSize, ColorLabel, FontStyle y Font	InputText.
<Image>	BackgroundImage	Panel, Button y Grid Layout Group.
<Padding>	PaddingTop, PaddingRight, PaddingLeft, PaddingBottom	Grid Layout Group.
<Cellsize>	CellSize_x, CellSize_y	Grid Layout Group.
<Spacing>	Spacing_x, Spacing_y	Grid Layout Group.

Teniendo en cuenta la Tabla 10, se presentan las siguientes reglas:

- Todo atributo y su posible valor debe ser en formato CamelCase.
- Cada propiedad va a ser una etiqueta y cada etiqueta va a tener un atributo llamado ***Key***.
- El atributo ***Key*** contiene un solo valor.

- Por cada valor que se pida del atributo ***Key***, debe haber una nueva etiqueta.
- Tanto la etiqueta como el valor de ***Key*** debe pertenecer al objeto (esto se puede ver reflejado en la tercera columna de la Tabla 10).
- No se aceptan espacios dentro del valor de las etiquetas, dentro de las mismas etiquetas ni dentro de atributos.

Por ejemplo, si queremos especificar valores para la etiqueta <Position>se especifican los posibles valores de ***Key***: *Left*, *Top*, *Right y Bottom*, entonces estos se representarán por cada etiqueta como se muestra en la figura 19.

```
<Position Key="Right"></Position>
<Position Key="Left"></Position>
<Position Key="Top"></Position>
<Position Key="Bottom"></Position>
```

Figura 19. Representación de los posibles valores de Key.

5.5.4.1.5 Regla de asignación de valores para las etiquetas.

Color

- Todo valor de la etiqueta color será en formato hexadecimal.
- Si el diseñador desea incluir opacidad al color, se hará concatenando el prefijo de la opacidad con el hexadecimal de la siguiente forma.

Posición

La posición de los objetos es jerárquica, esto quiere decir que es necesario tener los valores del objeto padre para poder distribuir los objetos de interfaz de usuario dentro de la ventana. En nuestro caso, el Canvas es quien reúne todos los objetos y cómo su Renderización pre- definida es de tipo Screen Space-Camera (Espacio de la pantalla-Cámara), se debe tener en cuenta las dimensiones (*Width y Height*) dadas por Unity para empezar a definir la posición y tamaño de los objetos que contiene. Por lo tanto, se definen las siguientes reglas:

- Las dimensiones iniciales del Canvas quien contiene todos los elementos de interfaz de usuario serán de [210 160], las cuales serán el punto inicial para definir la posición y tamaño de los elementos contenidos.
- La posición requiere cuatro propiedades: L*eft, Top, Right, Bottom* según el tipo de objeto que lo requiera (ver tabla 10).

La propiedad *Left* se define mediante la distancia del límite izquierdo de la ventana hasta el objeto.

La propiedad *Top* se define mediante la distancia del límite superior de la ventana hasta el objeto.

La propiedad *Right* se define mediante la distancia del límite derecho de la ventana hasta el objeto

La propiedad *Bottom* se define mediante la distancia del límite inferior de la ventana hasta el objeto.

El valor de la posición sólo se fija en números enteros.

Alineación o *Alignment*:

Esta regla definirá la alineación para el único objeto que la requiere, el *Label*. La alineación, define la posición vertical y horizontal del texto de la siguiente manera: *Lower*, *Middle* y *Upper* representa la posición vertical y *Center*, *Left* y *Right* representa la posición horizontal. Un claro ejemplo se puede observar en la siguiente figura:

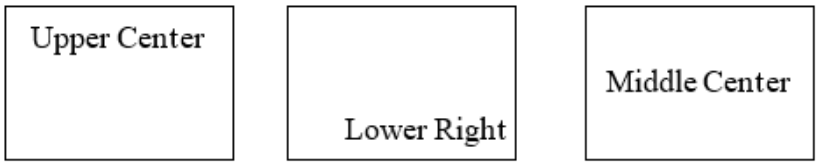


Figura 20. Ejemplo, Alineación de Texto en el Objeto Label.

Por lo cual se tiene la siguiente regla:

- El valor de Key será para la etiqueta <Alignment> puede contener uno de los nueve posibles valores (LowerCenter, LowerLeft, LowerRight, MiddleCenter, MiddleLeft, MiddleRight, UpperCenter, UpperLeft y UpperRight).

Imagen: Esta etiqueta es opcional, a continuación, se especifican sus reglas en caso de ser usada.

- El objeto imagen se especifica indicando en el valor de la etiqueta, la ruta donde está ubicada, todas las imágenes se ubicarán en la carpeta Images/.

Font: refiere a la tipografía del texto. Esta etiqueta es opcional, a continuación, se especifican sus reglas en caso de ser usada.

- La tipografía se especifica indicando en el valor de la etiqueta, la ruta donde está ubicada la tipografía, todas las tipografías se ubicarán dentro de la ruta Fuentes/Fonts/

FontStyle: se refiere al estilo de la tipografía, existen cuatro estilos (Normal, Bold, Italic y BoldAndItalic). Esta etiqueta es opcional; a continuación, se especifican sus reglas en caso de ser usada.

- El valor de Key para la etiqueta <FontStyle> puede contener uno de cuatro posibles valores.

FontSize, Padding, CellSize y Spacing:

- Estos valores solo se fijan en números enteros.

5.5.4.2 Construcción del archivo XML: A continuación, se definen los pasos básicos para realizar la construcción del archivo XML a partir del modelo de interfaz concreta, haciendo uso de las Reglas de concretización explicadas anteriormente.

- Como primera medida, es necesario identificar los objetos en Unity (esto para tener en cuenta que objeto poner en el atributo Type que muestra la Tabla 10).
- A continuación, se identifican los displays en la interfaz concreta.
- Posteriormente, se aplican las Reglas de concretización (“Regla de orden”, “Regla de selección” y “Regla de especificación de etiquetas”), estas reglas establecen la estructura general del archivo XML como se muestra en la Figura 21.

```
<Content Name=" Ingresar_Interfaz_Tutor" ID="0">
  <Display Name= "Seguir_Tutor" ID="0">
    <Object Type ="Panel" Name ="PanelTitle"></Object>
    <Object Type ="Panel" Name ="PanelContent"></Object>
    <Object Type ="Button" Name ="Next"></Object>
    <Object Type ="Button" Name ="Back"></Object>
    <Object Type ="Button" Name ="Close"></Object>
  </Display>
</Content>
```

Figura 21. Ejemplo, Estructura General de un Archivo XML.

- Luego, se aplica la regla “Regla de asignación de nombre de propiedades y atributos para los objetos” mediante la cual se agregan etiquetas de las propiedades de todos los objetos y sus atributos como se puede observar en la Figura 22.

```
<Content Name= "Ingresar_Interfaz_Tutor" ID="0">
  <Display Name= "Seguir_Tutor" ID="0">
    <Object Type ="Panel" Name ="PanelTitle">
      <Position Key="Right"></Position>
      <Position Key="Left"></Position>
      <Position Key="Top"></Position>
      <Position Key="Bottom"></Position>
      <Color Key="Normal"></Color>
      <Label Key="Alignment"></Label>
      <Label Key="FontSize"></Label>
      <Label Key="ColorLabel"></Label>
      <Label Key="FontStyle"></Label>
      <Label Key="Font"> </Label>
      <Image Key="BackgroundImage"> </Image>
    </Object>
    <Object Type ="Panel" Name ="PanelContent"></Object>
    <Object Type ="Button" Name ="Next"></Object>
    <Object Type ="Button" Name ="Back"></Object>
    <Object Type ="Button" Name ="Close"></Object>
  </Display>
</Content>
```

Figura 22. Ejemplo, Asignación de Nombres de Propiedades y Atributos para los Objetos.

- Finalmente, se aplica la regla “Regla de asignación de valores” con la cual se definen los valores a cada etiqueta; de esta manera se obtiene el archivo XML como se observa en la Figura 23.

```
<Content Name=" Ingresar_Interfaz_Tutor" ID="0">
  <Display Name= "Seguir_Tutor" ID= "0">
    <Object Type ="Panel" Name ="PanelTitle">
      <Position Key="Right">0</Position>
      <Position Key="Left">0</Position>
      <Position Key="Top">0</Position>
      <Position Key="Bottom">127</Position>
      <Color Key="Normal">#FFFFFF</Color>
      <Label Key="Alignment">MiddleCenter</Label>
      <Label Key="FontSize">15</Label>
      <Label Key="ColorLabel">#F3F3F3</Label>
      <Label Key="FontStyle">Italic</Label>
      <Label Key="Font">Fuentes/Fonts/nexa/Nexa-Bold</Label>
      <Image Key="BackgroundImage">Images/linea</Image>
    </Object>
    <Object Type ="Panel" Name ="PanelContent"></Object>
    <Object Type ="Button" Name ="Next"></Object>
    <Object Type ="Button" Name ="Back"></Object>
    <Object Type ="Button" Name ="Close"></Object>
  </Display>
</Content>
```

Figura 23. Ejemplo, Asignación de Valores a las Etiquetas.

Notas para tener en cuenta para la construcción del archivo XML.

- Existen propiedades **opcionales** que solo suman en la calidad del diseño.
- Es necesario ajustar todos los valores de color que tiene el objeto Button.
- Por cuestión de sintaxis y buena lectura del archivo XML, éste no acepta espacios dentro del valor de las etiquetas, dentro de las mismas etiquetas ni dentro de los mismos atributos.
- El valor del Canvas inicial para cualquier transformación siempre será el mostrado en la regla 5 de posición.
- Para la identificación del botón tiene dos opciones: una imagen a la cual se debe dar la ruta de la imagen o ponerle propiedades al Label del botón, para esta última opción debe especificar el texto (propiedad Text).
- Los nombres de los atributos, etiquetas y valores de los atributos son exactamente cómo se presentaron en este documento.
- Los nombres de los objetos son exactamente como se presentan en el modelo de interfaz concreto a transformar.
- Para mayor facilidad de la construcción del XML, use un editor de texto para código (como Sublime Text, Atom o Vscode).

5.6 Codificación e integración.

Esta fase se integra los archivos XML obtenidos para el diseño y se crea un archivo Json para el contenido; ambos requieren serializaciones para ser leídos dentro de Unity. También se especifican

demás integraciones que se consideraron útiles a partir de la generación de los archivos antes mencionados.

5.6.1 XML a Unity. Para la integración de XML a Unity, primero es necesario hacer la creación de los objetos definidos en la sección 4.5.4, esto debido a que en Unity es importante preservar la jerarquía para que cada objeto hijo herede características propias de su objeto padre. Luego a cada objeto se le asigna el script que administra los atributos *Key* como se observa en la Figura 24, para realizar la lista de atributos propios para cada objeto y poder llamar después los valores de dichos atributos.

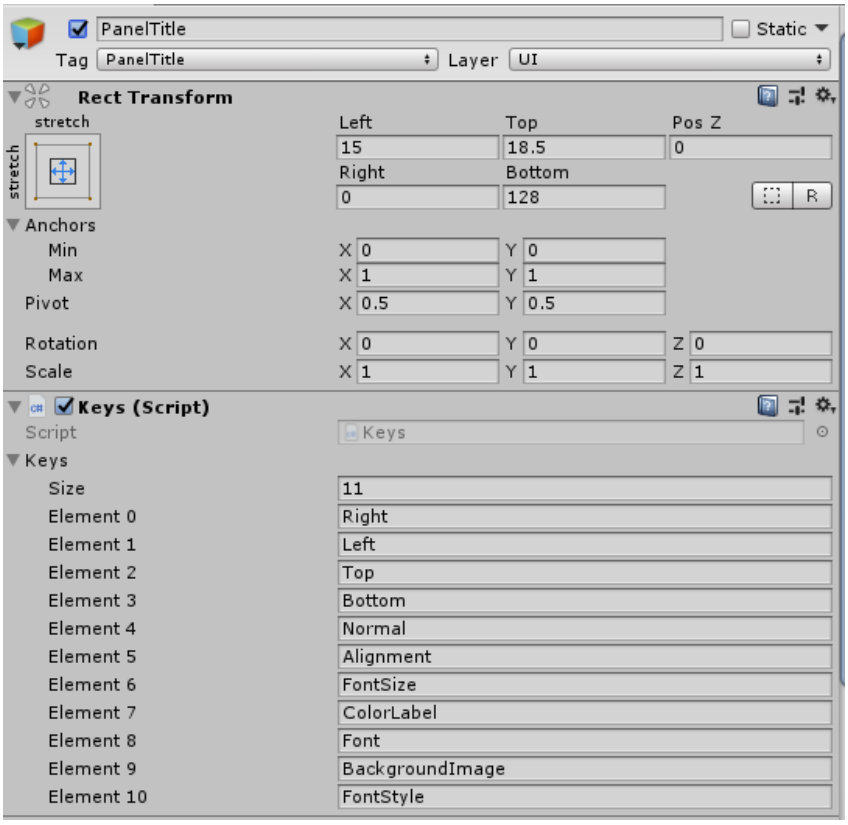


Figura 24. Especificación de Keys dentro del Inspector en el Entorno Unity.

Dentro de un script en C#, después se construye un método para Serializar el XML y agregarlo a una lista como se observa en la figura 25, la cual funcionará como un tipo de traductor para el programa que traerá los valores establecidos por el diseñador XML.

```
public void LoadXMLData()
{
    XmlSerializer serializer = new XmlSerializer(typeof(Content));
    FileStream stream = new FileStream(Application.dataPath + "/StreamingAssets/XML/CttIngresarEscenario.xml", FileMode.Open);
    FileStream stream2 = new FileStream(Application.dataPath + "/StreamingAssets/XML/CttEjecutarSolucionBasica.xml", FileMode.Open);
    FileStream stream3 = new FileStream(Application.dataPath + "/StreamingAssets/XML/CttDiagnosticarPega.xml", FileMode.Open);
    FileStream stream4 = new FileStream(Application.dataPath + "/StreamingAssets/XML/CttEjecutarSolucionEspecifica.xml", FileMode.Open);
    ItemsContent.Add(serializer.Deserialize(stream) as Content);
    ItemsContent.Add(serializer.Deserialize(stream2) as Content);
    ItemsContent.Add(serializer.Deserialize(stream3) as Content);
    ItemsContent.Add(serializer.Deserialize(stream4) as Content);
    stream.Close();
    stream2.Close();
    stream3.Close();
    stream4.Close();
}
```

Figura 25. Método que permite cargar los datos al entorno Unity.

Finalmente, se establece la ubicación de los archivos en la carpeta Streaming assets, que permite que al crear el ejecutable de la simulación, esta carpeta se mantenga con sus archivos y permita editar y modificar el diseño.

5.6.2 Json a Unity. Este paso se lleva a cabo teniendo en cuenta la estructura propuesta para el proceso de toma de decisiones establecida en la sección 4.1, donde se adapta la estructura que va a tener a lo largo de le ejecución de las soluciones para el problema operacional elegido.

Se escoge adaptar un archivo Json debido a que el contenido solo va a tener texto y valores booleanos, lo cual adaptarlo a este tipo de formato hace que nuestro trabajo tenga uso de nuevas tecnologías y además ligeras conforme a nuestro conjunto de datos.

Su estructura es conforme a las especificaciones de requisitos, por lo cual dentro del mismo archivo contiene un elemento llamado “allMessageRoundData” el cual se divide en niveles o

modulos a los que les llamamos “nameRound”, a su vez cada nivel o round, se divide en mensajes denominados “messages”, y cada mensaje contiene varios elementos que comprenden desde el mensaje que se va a mostrar “messageText”, hasta los botones que se desean mostrar con esa vista.

Este archivo se puede modificar y actualizar gracias a la función serialización presentada en XML, pero esta vez en contexto Json.

Se establece la ubicación de los archivos en la carpeta Streaming assets, que permite que al crear el ejecutable de la simulación, esta carpeta se mantenga con sus archivos y permita editar y modificar el contenido.

5.6.3 Otros tipos de integraciones. El resultado de las transformaciones representa las tareas y el diseño necesario para la realización de ayuda al proceso de toma de actividades definidas para el problema operacional presentado, sin embargo, pueden existir otros tipos de requerimientos que añaden mayor integración de la interfaz al simulador en cuestión. A partir de la creación de nuestro diseño, se definió un ítem que aporta al proceso:

Creación de botones que apuntan a cada pantalla de control del simulador: existen 10 pantallas para controlar el proceso de perforación, en las cuales se hace útil la navegación entre ellas dentro de la misma interfaz para la ejecución rápida de la solución al problema operacional. Es por esto, que se decide enlazar botones a dichas pantallas (por ejemplo, en la figura 26), lo cuales van apareciendo dentro de la interfaz, dependiendo del mensaje y la pantalla que se requiera observar.

Figura 26. Mensaje de la Interfaz que Apunta a dos Pantallas en el Proceso de Perforación.

5.7 Pruebas

En esta fase del proyecto se llevó a cabo el desarrollo de pruebas manuales, pruebas unitarias, pruebas de integración y pruebas con usuarios; mediante las cuales fue posible identificar posibles fallos; tanto en la aplicación de las Reglas de concretización de forma manual, como en el funcionamiento y cumplimiento de requisitos del prototipo de Interfaz de Usuario Tutor.

A continuación, se especifican las diferentes pruebas realizadas, así como los resultados obtenidos.

5.7.1 Prueba Manual Reglas de Trasformación. El desarrollo de esta prueba se llevó a cabo con 3 estudiantes graduados de Ingeniería de Sistemas y un estudiante de primer semestre de Ingeniería Electrónica. A cada evaluador se le brindo formación previa acerca de los diferentes modelos; posteriormente, cada evaluador procedió a hacer la transformación de cada uno de los modelos haciendo uso de las reglas de concretización expuestas en este documento.

5.7.1.1 Prueba Manual para Reglas de concretización de modelo de tareas a modelo de interfaz abstracta. A continuación, se identifican los tipos de reglas aplicadas en la transformación de modelo de tareas a modelo de interfaz abstracta.

- Regla 1: Es la “Regla de orden”, esta regla se compone de 2 reglas.
- Regla 2: Es la “Regla de identificación”, esta regla se compone de 8 reglas.
- Regla 3: Es la “Regla de posición”, esta regla se compone de 4 reglas.
- Regla 4: Es la “Regla de concretización”, esta regla se compone de en 9 reglas

Dado que las Reglas de concretización a aplicar son las mismas para todos los modelos de esta transformación, se generó una prueba estándar, donde se enseña el número de reglas correctas sobre el total de reglas que componen cada tipo.

Tabla 11. *Resultados Prueba de Transformación Modelo de Tareas a Modelo de Interfaz Abstracta.*

Evaluador	Transformación Modelo de Tareas a Modelo de Interfaz Abstracta				
	Regla 1	Regla 2	Regla 3	Regla 4	Promedio
1	2/2	6/8	4/4	8/9	0,9097
2	2/2	6/8	4/4	8/9	0,9097
3	2/2	6/8	4/4	7/9	0,8819
4	2/2	6/8	4/4	8/9	0,9097
Porcentaje de efectividad					0,9027 = 90,27%

En la tabla 11 se observan los resultados de la aplicación de esta prueba para cada evaluador, haciendo un promedio se obtuvo que se presenta una efectividad en la aplicación de la prueba del 90,27%; por lo tanto, un error del 7,3%. Al analizar los motivos a los cuales se debe este error se evidencia que es debido a que el evaluador tiende a generalizar las reglas omitiendo aquellas que son específicas según el caso.

5.7.1.2 Prueba Manual para Reglas de concretización de modelo de interfaz abstracto a modelo de interfaz concreto: A continuación, se identifican los tipos de reglas aplicadas en la transformación de modelo de interfaz abstracta a modelo de interfaz concreta.

- Regla 1: Es la “Regla de selección”, esta regla se compone de 6 reglas.
- Regla 2: Es la “Regla de enlace”, esta regla se subdivide en 4 reglas.
- Regla 3: Es la “Regla de identificación”.

Dado que las Reglas de concretización a aplicar son las mismas para todos los modelos, se generó una prueba estándar, donde se enseña el número de reglas correctas sobre el total de reglas en las que se subdivide cada tipo.

Tabla 12. *Resultados prueba de Transformación Modelo de Interfaz Abstracto a Modelo de Interfaz Concreto.*

Evaluador	**Transformación Modelo de Interfaz Abstracto a Modelo de Interfaz Concreto**			
	Regla 1	Regla 2	Regla 3	Promedio
1	5/6	4/4	1	0,9444
2	4/6	4/4	1	0,8888
3	6/6	4/4	1	1
4	6/6	4/4	1	1
Porcentaje de Efectividad				0,9583 = 95,83%

En la tabla 12 se observan los resultados de la aplicación de esta prueba para cada evaluador, haciendo un promedio se obtuvo que se presenta una efectividad en la aplicación de la prueba del 95,83%; por lo tanto, un error del 4,17%. Al analizar los motivos a los cuales se debe este error se evidencia que es debido a que el evaluador tiende a generalizar las reglas omitiendo aquellas que son específicas según el caso.

5.7.2 Prueba Unitaria, creación de archivo XML a partir del modelo de interfaz concreto.

El desarrollo de esta prueba se llevó a cabo con los mismos tres estudiantes graduados de Ingeniería de Sistemas y el estudiante de primer semestre de Ingeniería Electrónica.

A cada evaluador se le brindo formación previa acerca de los diferentes objetos que se prestan en el entorno Unity así como sus respectivas propiedades, además de una instrucción básica sobre la creación de un archivo XML; posteriormente, cada evaluador procedió a crear el archivo XML de un modelo en específico (el mismo para los cuatro evaluadores); en esta prueba, se partió del modelo de interfaz concreta para la ejecución de la solución básica y se generó el archivo XML haciendo uso de las reglas expuestas en la sección 4.5.4.

A continuación, se identifican los tipos de reglas aplicadas en la creación de un archivo XML a partir del modelo de interfaz concreta y así obtener el modelo de interfaz final.

Regla 1: Es la “Regla de orden”.

Regla2: Es la “Regla de selección”, esta se compone de seis reglas.

Regla 3: Es la “Regla de especificación de atributos para cada etiqueta”, esta se compone de seis reglas.

Regla 4: El la “Regla de asignación del nombre de las propiedades y los posibles calores de Key para cada objeto”, esta regla se compone de seis reglas.

Regla 5: Es la “Regla de asignación de valores”, esta se compone de nueve reglas, dentro de estas nueve reglas se tienen 3 opcionales; por la tanto, se evaluará sobre el número de reglas aplicadas por cada diseñador.

En los resultados de la prueba se enseña, el número de reglas correctas sobre el total de reglas en las que se subdivide cada tipo; para el caso de la regla 5, se señala de reglas correctas sobre el número de reglas aplicadas.

Tabla 13. *Resultados de Transformación de Modelo de Interfaz Concreto a Modelo de Interfaz Final.*

Evaluador	**Transformación de Modelo de Interfaz Concreto a Modelo de Interfaz Final**					
	Regla 1	Regla 2	Regla 3	Regla 4	Regla 5	Promedio
1	1	6/6	6/6	4/6	7/7	0,9333
2	1	6/6	6/6	5/6	7/7	0,9666
3	1	6/6	6/6	4/6	7/7	0,9333
4	1	6/6	6/6	4/6	7/7	0,9333
5	1	6/6	6/6	6/6	6/7	0,9714
6	1	6/6	6/6	6/6	7/7	1
7	1	6/6	6/6	4/6	7/7	0,9333
8	1	6/6	6/6	5/6	6/7	0,9381
9	1	6/6	6/6	6/6	5/7	0,9429
10	1	6/6	6/6	5/6	6/7	0,9381
Porcentaje de efectividad						0,9490 = 94,90%

En la tabla 13 se observan los resultados de la aplicación de esta prueba para cada evaluador, haciendo un promedio se obtuvo que se presenta una efectividad en la aplicación de la prueba del 94,90%; por lo tanto, un error del 5,71%. Al analizar los motivos a los cuales se debe este porcentaje de error se determinó que puede ser debido a la inexperiencia por parte del evaluador al hacer archivos XML, así como falta de comprensión u omisión de algunas reglas dadas.

5.7.3 Prueba de Integración

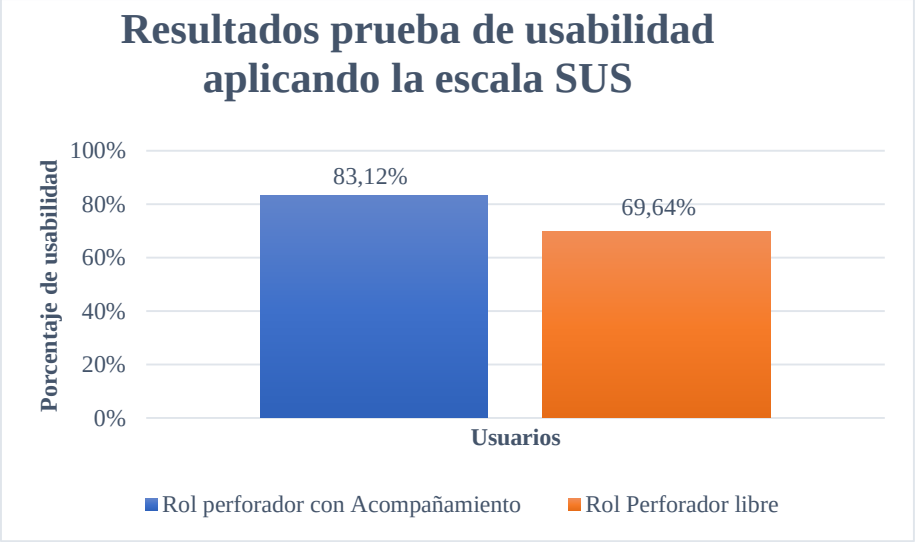
Figura 27. Integración de diferentes archivos XML al entorno Unity

Para el desarrollo de esta prueba, se tuvieron en cuenta diez archivos XML entre estos se encuentran los generados por los evaluadores de la prueba unitaria; lo que se pretendía con esta prueba, era mostrar la integración del archivo XML al entorno Unity; por lo tanto, en la figura 27 se muestran algunos de diseños obtenidos dónde se evidencia el cambio tanto en el diseño (posición de los objetos, color de los objetos, tamaño de letra, tipo de objetos, entre otros) como en el contenido.

5.7.4 Pruebas con usuarios. La prueba se lleva a cabo con 15 estudiantes de sexto nivel de Ingeniería de Petróleos de la Universidad Industrial de Santander, se realizó en cuatro sesiones, tres sesiones conformadas por cuatro estudiantes y una por tres. En el desarrollo de cada sesión, se cuenta con dos estudiantes que cumplen el rol de perforador con acompañamiento y dos que cumplen con el rol de perforador libre.

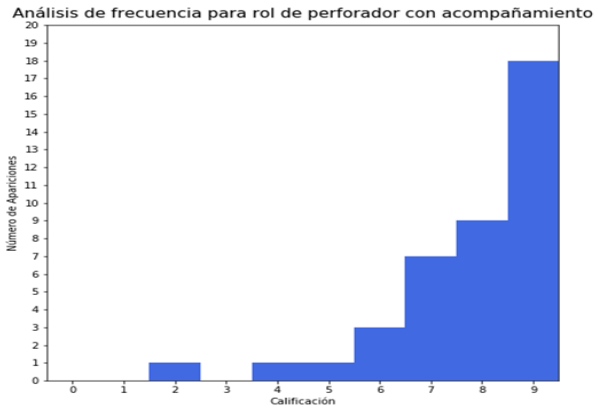
Para evaluar la prueba, se evalúa al estudiante bajo métricas establecidas (Tiempo en realizar la tarea, número de errores y número de solicitudes de ayuda) a lo largo de la misma y al finalizar la prueba se le solicita a cada estudiante responder el cuestionario SUS, este cuestionario es el Sistema de Escalas de Usabilidad o simplemente SUS mediante el cual se mide la usabilidad del simulador tanto haciendo uso de la interfaz Tutor, como sin hacer uso de la misma. El formato mediante el cual se evaluaron las métricas se puede ver en el apéndice H.

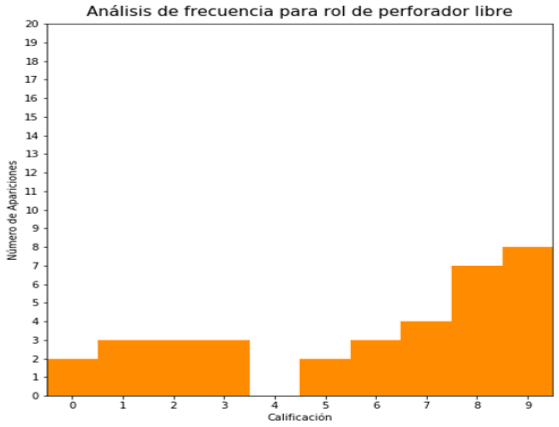
A continuación, se presentan los resultados obtenidos con la aplicación de esta prueba.



En el grafico se aprecia el resultado de la aplicación de la escala SUS a los estudiantes luego de haber realizado la prueba, como se evidencia, se tiene un mayor porcentaje de usabilidad del simulador al hacer uso de la interfaz Tutor

A continuación, se representa mediante un histograma las reacciones globales de los estudiantes al usar el simulador para los dos roles establecidos anteriormente. Estas reacciones globales se evalúan mediante la frecuencia de las impresiones del usuario, que van desde cero hasta nueve, iniciando en cero con apreciaciones negativas (Terrible, frustante, aburrido difícil y rigido) y finalizando en nueve con apreciaciones positivas (Agradable, satisfactorio, estimulante, fácil y flexible).





Como se observa en el histograma, hay más presencia de apreciaciones negativas en los estudiantes que cumplían el rol de “perforador libre” en comparación con los estudiantes que cumplían el rol de “perforador con acompañamiento” es decir, se tienen más apreciaciones negativas al hacer uso del simulador sin la interfaz de usuario tutor.

Mediante la siguiente tabla se representan los resultados de las métricas aplicadas a cada estudiante durante la aplicación de la prueba de usabilidad.

Métricas evaluadas	Rol perforador con acompañamiento	Rol Perforador Libre
Tiempo (min).	14,125	26,428
Número de errores.	7	4
Número de solicitudes de ayuda.	13	11
% Estudiantes con tarea finalizada al primer intento.	75%	28,57%

En esta evaluación, se puede evidenciar como disminuye el tiempo de entrenamiento de los aprendices al hacer uso de la interfaz de usuario Tutor. Así mismo, es posible para el instructor llevar un mayor control en las diferentes falencias que presentan los aprendices ya que se brinda un entrenamiento con información procedimental por medio de el modelo instruccional basado en la teoría cognitivista mediante el cual se estructura el entrenamiento por niveles lo cual facilita evidenciar tanto las fortalezas como las falencias específicas de cada aprendiz. Igualmente se evidencia cómo el aprendiz acude al instructor por ayuda dado que al ser guiado en el proceso le permite tener un mayor análisis de la situación y considera necesario que el instructor le despeje dudas. Finalmente se evidencia una alta diferencia porcentual entre los usuarios que finalizan la tarea en un intento en comparación con los que no lo logran al primer intento. Deducimos, que esto se debe a que, al no ser guiados los estudiantes con “rol perforador”, no brindan una solución correcta al problema operacional presentado, además de manipular inadecuadamente el simulador, por lo tanto, fue necesario un segundo intento con la colaboración de sus compañeros que cumplieron el Rol perforador con acompañamiento o instructor para dar solución al problema operacional presentado.

El protocolo de esta prueba puede verse en el apéndice G.

6. Análisis y resultados

A continuación, se presenta el análisis detallado de los resultados obtenidos de las diferentes pruebas realizadas y expuestas en el capítulo anterior.

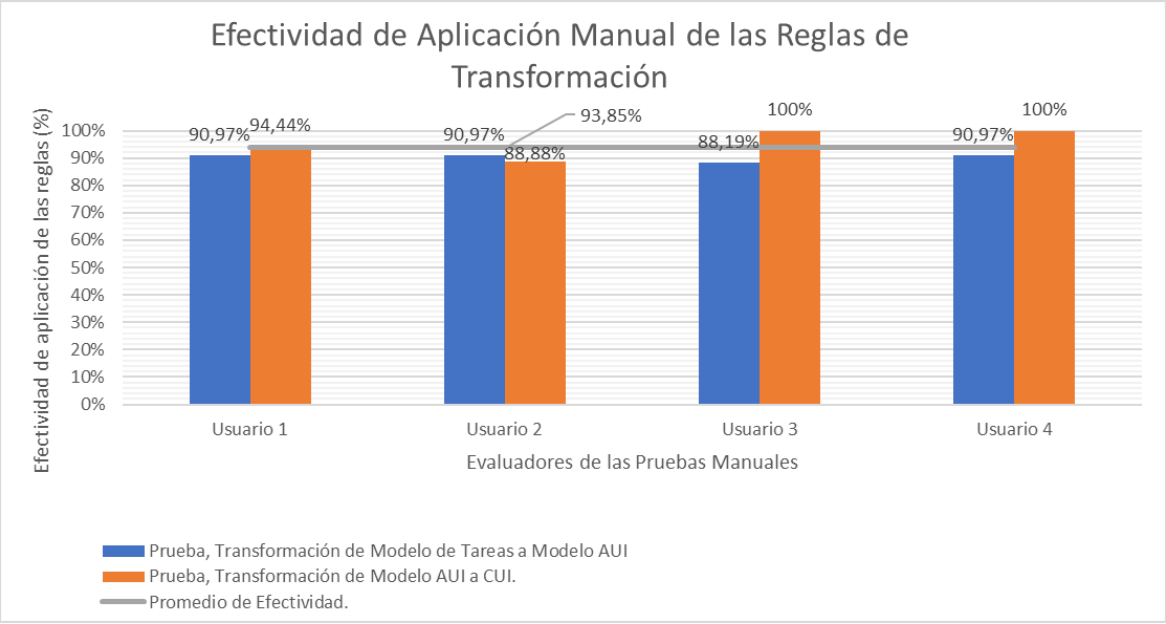


Figura 28. Efectividad de Aplicación Manual de las Reglas de Concretización.

En la figura 28 se tiene el promedio de efectividad en la ejecución de las pruebas manuales por cada evaluador, se puede observar que disminuyo el porcentaje de error al aplicar la prueba "aplicación de las reglas de concretización del modelo AUI al modelo CUI", en comparación con la prueba de "aplicación de las reglas de concretización del modelo de tareas al modelo AUI". Consideramos que se debe a que al ser la segunda prueba aplicada; el evaluador ya se ha familiarizado con los diferentes modelos y ha identificado la necesidad e importancia de atender

correctamente las reglas planteadas para la transformación de cada modelo. Finalmente, en la figura 28 se pude apreciar que el promedio de efectividad de esta prueba es del 93,85%, considerando este promedio como un alto nivel de efectividad de aplicación de las reglas de concretización, teniendo en cuenta que fueron aplicadas por personas naturales, susceptibles a errores.

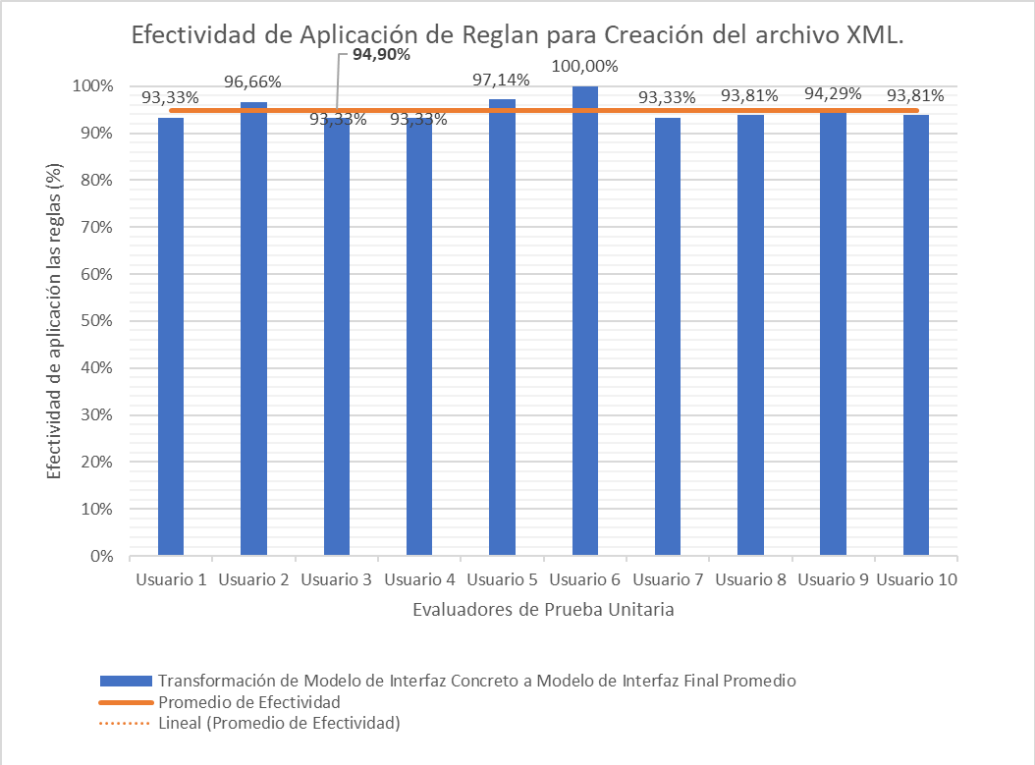


Figura 29. Efectividad de Aplicación de las Reglas para la creación de un archivo XML.

En la figura 29, se observa el promedio de efectividad en la ejecución de las pruebas unitarias, mediante estas pruebas se evaluó la efectividad de la creación del archivo XML, teniendo como resultado un promedio de efectividad del 94,9%, considerando este promedio como un alto nivel

de efectividad, teniendo en cuenta que fueron aplicadas por personas naturales, susceptibles a errores.

Debido a que se tuvo mayor cantidad de datos, se decidió hacer un análisis más profundo aplicando análisis de concordancias y teorema de Bayes.

Análisis de concordancias

Se realiza un análisis de concordancia para evaluar la uniformidad y exactitud de aplicación de las reglas por parte de los evaluadores en comparación al estándar que representa el ideal para cada regla.

En donde se obtienen los siguientes resultados:

1) Exactitud general: para evaluar las coincidencias con el estándar.

$$100 * \frac{X}{N}$$

X= Numero de evaluaciones que coinciden con el estándar

N= Numero de filas de datos válidos

$$100 * \frac{38}{50} = 76\% \; de \; exactitud \; general$$

2) Exactitud para cada evaluador: mide la concordancia de cada evaluador con el estándar.

$$100 * \frac{Numero \; de \; evaluaciones \; que \; coinciden \; con \; el \; valor \; estándar}{N_i}$$

Siendo N_i el número de evaluaciones.

Cada evaluador vs. el estándar

Acuerdo de evaluación

Evaluador	No. de inspeccionados	No. de coincidencias	Porcentaje
1	5	4	80,00
2	5	4	80,00
3	5	3	60,00
4	5	4	80,00
5	5	4	80,00
6	5	5	100,00
7	5	4	80,00
8	5	3	60,00
9	5	4	80,00
10	5	3	60,00

Figura 30. Tabla con datos evaluador vs estándar extraído de Minitab.

Análisis aplicando teorema de Bayes

Se extrae un árbol que se observa en la figura 31 que representa los datos obtenidos dentro de los evaluadores por cada regla lo que representan las probabilidades a priori.

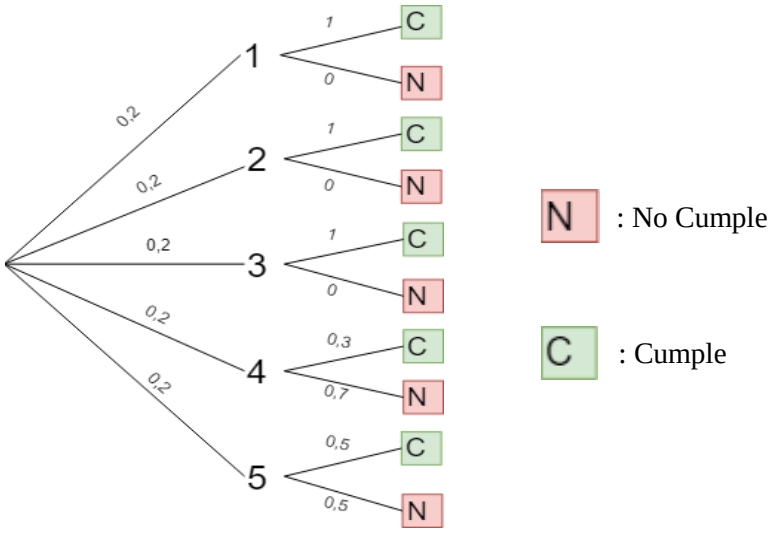


Figura 31. Árbol que representa los datos obtenidos en la prueba unitaria.

Los números 1,2,3,4 y 5 representan las reglas, y todas tienen la misma probabilidad porque todos los evaluadores hicieron las mismas reglas, además se observan las probabilidades de que si alguna regla cumple o no cumple.

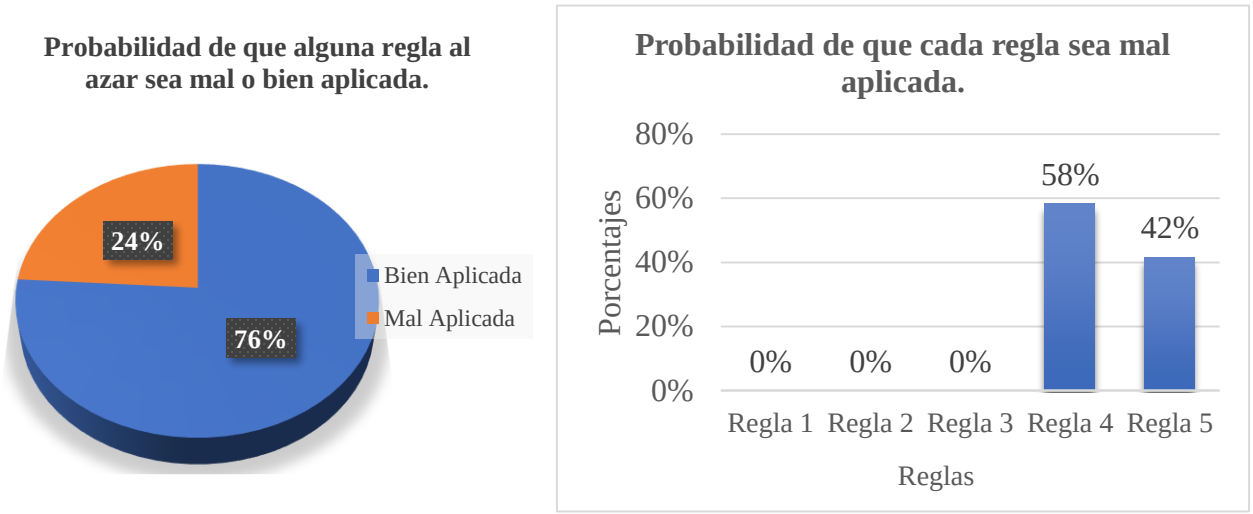


Figura 32. Resultados obtenidos aplicando teorema de Bayes.

En la figura 32 podemos observar los resultados obtenidos a dos probabilidades evaluadas con el teorema de Bayes.

De las pruebas unitarias y los análisis para los datos realizados podemos concluir que existe una exactitud general del 76% de que una regla sea correctamente aplicada, además que hay mayor probabilidad que la regla número cuatro de asignación del nombre de las propiedades y los posibles valores de Key para cada objeto sea incorrectamente aplicada, deducimos que esto sucede debido a que se debe asignar estructuras específicas y establecidas dentro de la misma regla que muchos de los usuarios suelen omitir.

A partir de un formato de cumplimiento de requerimientos, valorado por 10 ingenieros de sistemas graduados, fue posible identificar que, en el desarrollo de este proyecto, dichos requerimientos se cumplen en su totalidad.

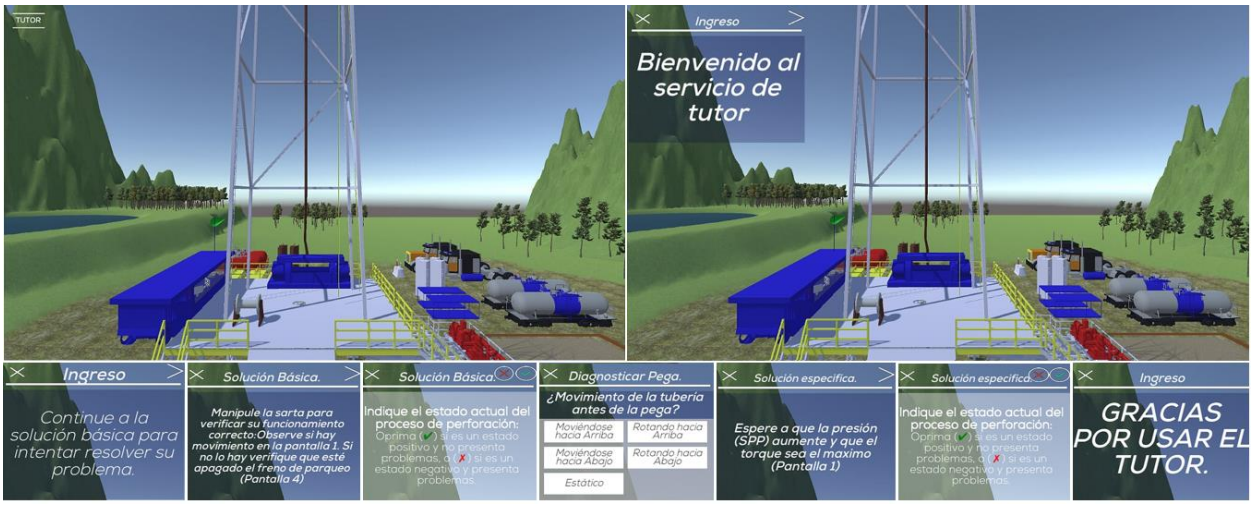


Figura 33 Diseño y Contenido de la Interfaz de Usuario Tutor.

En la Figura 33 se aprecia el diseño y contenido de la Interfaz de usuario tutor, el diseño se integró a la misma mediante un archivo XML y su contenido, mediante un archivo Json.

También se puede observar la integración de dicha interfaz dentro del simulador preliminar a este proyecto, cabe recordar, que este simulador fue un proyecto desarrollado por estudiantes de ingeniería de petróleos; nuestro trabajo fue integrar a este simulador, una interfaz de usuario Tutor haciendo uso del enfoque de Diseño de Interfaz de Usuario Basado en Modelos, MBUID.

La interfaz de usuario Tutor se puede ver por completo en el en el apéndice E.

Finalmente, mediante la prueba de usabilidad realizada con estudiantes de Ingeniería de Petróleos se pudo evidenciar que la integración de la interfaz de usuario Tutor en el entorno de simulación tuvo una buena acogida por parte de los estudiantes, mostrándose una usabilidad mayor, unas mejores apreciaciones globales, un menor tiempo en el desarrollo de solución al problema operacional, además de que fomenta el análisis de la situación en los estudiantes y permite al instructor evidenciar con mayor facilidad y certeza las diferentes fortalezas y falencias

que presenta cada aprendiz, todo esto comparado con el uso del entorno de simulación sin la interfaz integrada.

7. Conclusiones y trabajo a futuro

- Se definió el conjunto de actividades de toma de decisiones necesarias para dar solución a un problema operacional, específicamente el problema operacional pega por empaquetamiento, a través de la construcción de un diseño instruccional basado en una teoría cognitivista con información procedimental algorítmica, para esto se despliega de manera jerárquica cuatro niveles de ejecución de la solución dando la posibilidad al aprendiz de adquirir habilidades mentales que le permitan establecer criterios para poner en práctica los nuevos conocimientos, así como desarrollar la capacidad de tomar decisiones.
- Mediante la aplicación del enfoque MBUID (Model Based User Interface Development) y el framework Cameleon fue posible realizar la transformación de un modelo con alto nivel de abstracción (Modelo de Tareas o de dominio) a otro con un nivel de menor abstracción (Modelo FUI), haciendo uso de las reglas de concretización establecidas en este documento y como resultado obtener el diseño de la interfaz de usuario final dependiente de la plataforma.
- El desarrollar una interfaz de usuario usando el enfoque MBUID y el framework Cameleon, permite obtener una interfaz con contenido y diseño flexible que se puede acoplar a diferentes tipos de plataformas de desarrollo, así como separar o integrar la interfaz obtenida de forma escalonada a diferentes tipos de software. En este proyecto, el diseño de la interfaz de usuario

Tutor se representa mediante el lenguaje de marcado XML y su contenido se representa mediante un archivo de texto Json. Estos archivos se integraron a un proyecto preliminar desarrollado en la plataforma Unity realizado por estudiantes de la Escuela de Ingeniería de Petróleos. La unión de XML y Json permite optimizar el desarrollo de la interfaz ya que a medida que cambia el diseño mediante XML, paralelamente está cambiando el contenido de este mediante Json. Para integrar los archivos XML y Json a la plataforma Unity se realizó una serialización a través del lenguaje C# quien hace el papel de traductor entre estos archivos y Unity, permitiendo su modificación y actualización de manera inmediata.

- Según las pruebas aplicadas, se considera que el uso de este enfoque es efectivo, puesto que, aunque se necesita una revisión constante y reiterada de la documentación, además de una buena interpretación del mismo para crear las reglas de concretización apropiadas, se obtiene que la transformación de un modelo a otro mediante reglas de concretización, genera una efectividad superior al 90%.
- Se identificó la necesidad de realizar un proceso de formación preliminar con los evaluadores de manera que la aplicación de las reglas sea completamente efectiva ya que los errores observados durante la aplicación de las pruebas se debían en gran medida, por falta de comprensión, concentración u omisión por parte de los evaluadores a la hora de aplicar la prueba.
- La estructura de la interfaz tutor fomenta el análisis de la situación a los estudiantes, además de facilitar al instructor evidenciar tanto las fortalezas y la correcta toma de decisiones como las falencias en áreas específicas del aprendiz, lo cual hace posible brindar un entrenamiento más guiado y enfocado a las necesidades de cada aprendiz.
- Como trabajo a futuro se puede considerar el desarrollo de una herramienta que permita sistematizar las reglas de concretización, estructurándolas de tal forma que su entrada sean

modelos de tareas, y como salida se obtengan lo modelos abstractos y concretos automáticamente, además de generar el archivo XML el cual se implementa la interfaz de usuario final.

Referencias Bibliográficas

abialeba. (2014). Blog semana 4 Ing. Software I.

Abraham, M. M. (2007). *Especificación de modelos de tareas a partir de especificaciones de interfaces de usuario.*

Alarcon, A. (2014). Guía rápida de las principales teorías del diseño instruccional. Recuperado el 29 de julio de 2019, de https://www.shiftelearning.com/blogshift/bid/345775/Una-gu-a-r-pida-de-las-principales-teor-as-del-dise-o-instruccional-Infograf-a

Aroca, Á. (2012). Unity 3D, desarrollo de videojuegos para iOS y Android, gratis hasta el 8 de Abril. Recuperado el 3 de abril de 2018, de https://www.genbetadev.com/herramientas/unity-3d-desarrollo-de-videojuegos-para-ios-y-android-gratis-hasta-el-8-de-abril. [Accessed: 03-Apr-2018]

Belloch, C. (2012). *Diseño Instruccional*. Recuperado de https://www.uv.es/bellochc/pedagogia/EVA4.pdf

Blanca Azahara, A. S. (2007). *coCTT: Modelado de tareas en un entorno colaborativo*.

Bottoni, P., Borgia, F., Buccarella, D., Capuano, D., De Marsico, M., & Labella, A. (2013). Introduction to Model-Based User Interfaces. *Universal Access in the Information Society*. https://doi.org/10.1007/s10209-012-0283-y

Candil, D. (2014). Unity, el motor de desarrollo capaz de partir la historia de los videojuegos en dos. Recuperado el 3 de abril de 2018, de https://www.vidaextra.com/industria/unity-el-motor-de-desarrollo-capaz-de-partir-la-historia-de-los-videojuegos-en-dos

Contenidos conceptuales, procedimentales y actitudinales. (2009). Recuperado de

http://ideascompilativas.blogspot.com/2009/06/contenidos-conceptuales-procedimentales.html

Coordinación Ojo-Mano u Óculo-Manual - Habilidad Cognitiva. (s/f). Recuperado el 3 de abril de 2018, de https://www.cognifit.com/es/habilidad-cognitiva/coordinacion-ojo-mano.

Crockford, D. (s/f). Introducción a JSON.

Definición de XML - Qué es, Significado y Concepto. (s/f).

Descripción General | e-Tech Simulation. (s/f).

Fernandez Ruiz, M. J., Angós Ullate, J. M., & Salvador Olivan, J. A. (2001). *Dialnet-InterfacesDeUsuario-1456152*.

Gomez, M. M. (2018). Modelos de Diseño Instruccional que debes conocer. Recuperado el 29 de julio de 2019, de http://elearningmasters.galileo.edu/2018/10/03/modelos-diseno-instruccional/

Importancia de la toma de decisiones. (2008). Recuperado de https://www.xuletas.es/ficha/importancia-de-la-toma-de-decisiones/.

Interaccion entre Humanos y Computadoras: Tendencias Actuales y Futuras Introduccion del Editor Invitado | Septiembre 2014 | Computing Now - IEEECS. (s/f).

Interfaz de usuario - EcuRed. (s/f).

Jaime, M. A., Yosly Caridad, H., Viviana, B. A., Chavarria, A. A., Calderon, M. E., Collazos, C. E., … Rosa, G. (2014). *Temas de diseño en Interacción Humano - Computadora*.

León, F. (2012). *Contenidos Procedimentales*. Recuperado de https://es.slideshare.net/leonfdiaz/contenidos-procedimentales-13229831

Martinez, C. (2014). Habilidad para la Toma de Decisiones. Recuperado el 3 de abril de 2018, de https://www.eoi.es/blogs/mintecon/2014/05/18/habilidad-para-la-toma-de-decisiones/

Materiales Unity. (2018). *3*. Recuperado de https://docs.unity3d.com/es/current/Manual/Materials.html

Muñoz Márquez, F. J. (2007). *ACAUI: Abstracción de interfaces de usuario a*.

Pamela, C. (s/f-a). 2.3 Objetivos - Seminario I. Recuperado de https://sites.google.com/site/seminario1les/contenido-de-la-asignatura/Unidad-N2/2-La-relacion-Hombre-Mquina-y-su-esquematizacion/23-Objetivos

Pamela, C. (s/f-b). Definición de Interacción Hombre-Máquina - Seminario I. Recuperado el 3 de abril de 2018, de https://sites.google.com/site/seminario1les/contenido-de-la-asignatura/Unidad-N2/2-La-relacion-Hombre-Mquina-y-su-esquematizacion/21-Definicin-de-Interaccion-Hombre-Maquina

Pérdida de Circulación (Ocurrencia, Detección, Prevención y Soluciones). (s/f).

Perforación. (s/f).

Problemas comunes de perforación relacionados con (1). (s/f).

Quijada, C., Caraballo, H., & Garcias, A. (2011). *Analisis de Los Problemas Operacionales Durante La Perforacion de Pozos Petroleros*. Recuperado de https://es.scribd.com/document/334034959/Analisis-de-Los-Problemas-Operacionales-Durante-La-Perforacion-de-Pozos-Petroleros

Ramírez-Acosta, K. (s/f). *Interfaz y experiencia de usuario: parámetros importantes para un diseño efectivo User experience and user interface: important parameters for an effective design*. https://doi.org/10.18845/tm.v30i5.3223

Sanchez, J. (s/f). JSON: ¿Qué es JSON?, ¿Cómo se usa? y ¿Para qué sirve JSON?

Toinga, L. (2012). *Escuela politécnica nacional*. 150.

Un buen diseño de la interfaz de usuario en el éxito de aplicaciones de software. (s/f).

XML ¿QUE ES? (s/f)