INSTITUTO TECNOLÓGICO DE IZTAPALAPA

INGENIERÍA EN SISTEMAS COMPUTACIONALES

Propuesta para el desarrollo del proyecto:

LÓGICA COMBINATORIA

Presenta:

ARREDONDO FLORES ALEXA KETZALI
BOHORQUEZ LOPEZ MIGUEL ANGEL
RODRIGUEZ GARCIA JESUS

No. De Control:

181080005
181080006
181080027

Asesor Interno:

PARRA HERNANDEZ ABIEL TOMAS

CIUDAD DE MÉXICO **NOVIEMBRE / 2020**

LÓGICA COMBINATORIA

Resumen

La lógica combinatoria (Combinatory Logic) es una teoría lógica que está conectada a muchas áreas de la lógica y que ha encontrado aplicaciones en otras disciplinas, especialmente, en informática y matemáticas.

En este proyecto se verán algunos ejemplos que utilizan esta lógica, alguno de esos ejemplos son “La construcción de Chaitin”.

Con este proyecto se podrá entender mejor cómo funciona la lógica combinatoria, como también un poco de su historia.

Palabras clave: Lógica combinatoria, matemáticas, informática.

Cronograma

Instituto Tecnologico de Iztapalapa
Cronograma del Proyecto “ Lógica Combinatoria”

Actividad	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Recopilación de la información sobre la investigación																
Construcción del resumen y cronograma																
Lluvia de ideas con toda la información recabada																
Construcción del desarrollo del proyecto																
Análisis de Riesgos																
Descripción del modelo																
Investigación sobre el problema de Halting																
Hipótesis y Propuesta																
Investigación sobre la metodología a usar																
Requerimientos Funcionales y no Funcionales																
Diseño de Actividades y Proyecto																
Programa a realizar																
Entrega del proyecto de investigación																

Análisis de Riesgos

RIESGO	SOLUCIÓN
1.- Falta de conocimiento y comprensión del tema.	El equipo se tomará el tiempo necesario en investigaciones para el buen entendimiento del tema.
2.- Discrepancia y conflicto entre los miembros del equipo.	Ejecutar dinámicas de grupo para fortalecer el trabajo en equipo y el compañerismo.
3.- Falta de conocimiento y comprensión del software "JFLAP" .	Todo el equipo debe participar y tratar de trabajar con el software para su entendimiento.
4.- Malas estimaciones de tiempo (Cronograma de trabajo poco real).	El equipo debe comprometerse con el proyecto, se deberá seguir el cronograma con el tiempo especificado de trabajo.
5.- Falta de actividades del Cronograma.	Conforme se vaya avanzando en la investigación, se irán agregando las actividades pertinentes.

Descripción del modelo

La lógica combinatoria es una notación para eliminar la necesidad de variables cuantificadas en lógica matemática. Fue introducida por Moses Schönfinkel y Haskell Curry, se basa en combinadores que fueron introducidos por Schönfinkel en 1920 con la idea de proporcionar una forma análoga de construir funciones y eliminar cualquier mención de variables, particularmente en la lógica de predicados.

El objetivo de la lógica combinatoria era comprender mejor las paradojas y establecer conceptos matemáticos fundamentales sobre principios más simples y limpios que los marcos matemáticos existentes, especialmente para comprender mejor el concepto de *sustitución*.

La lógica combinatoria tiene gran importancia en los fundamentos de las matemáticas o la informática.

En matemáticas, la lógica combinatoria se pensó originalmente como una "pre-lógica" que declararía el papel de las variables cuantificadas en la lógica, esencialmente eliminandolas.

En informática, la lógica combinatoria se utiliza como un modelo simplificado de cálculo, utilizado en la teoría de la computabilidad y la teoría de la prueba. A pesar de su simplicidad, la lógica combinatoria captura características esenciales de la computación.

JUSTIFICACIÓN

Problema de "Halting"

En la teoría de la computabilidad , el problema de la parada o "Halting problem" es el problema de determinar, a partir de una descripción de un programa de computadora arbitrario y una entrada, si el programa terminará de ejecutarse o continuará ejecutándose infinitamente. Alan Turing demostró en 1936 que no puede existir un algoritmo general para resolver el problema de detención para todos los posibles pares de entrada de programa, es decir, es indecidible (no computable o no recursivo), en el sentido de que ninguna máquina de Turing lo puede resolver.

Para cualquier programa "f" que pueda determinar si los programas se detienen, un programa "patológico" "g", llamado con alguna entrada, puede pasar su propia fuente y su entrada a "f" y luego hacer específicamente lo contrario de lo que "f" predice que "g" hará. No puede existir ninguna persona que maneje este caso. Una parte clave de la demostración es una definición matemática de una computadora y un programa, que se conoce como máquina de Turing; el problema de la detención o "Halting" es indecidible en las máquinas de Turing. Es uno de los primeros casos de problemas de decisión que han demostrado ser irresolubles. Esta prueba es significativa para los esfuerzos prácticos de computación, ya que define una clase de aplicaciones que ninguna invención de programación puede realizar perfectamente.

Un poco de Historia

El problema de la detención es históricamente importante porque fue uno de los primeros problemas en ser indecidible. (La prueba de Turing se imprimió en mayo de 1936, mientras que la prueba de Alonzo Church de la indecidibilidad de un problema en el cálculo lambda ya se había publicado en abril de 1936 [Church, 1936]). Posteriormente, se han descrito muchos otros problemas indecidibles.:

Cronología

- 1900: David Hilbert plantea sus "23 preguntas" (ahora conocidas como problemas de Hilbert) en el Segundo Congreso Internacional de Matemáticos en París. “De estos, el segundo fue el de probar la consistencia de los 'axiomas de Peano' de los que, como había demostrado, dependía el rigor de las matemáticas”. (Hodges p. 83, comentario de Davis en Davis, 1965, p. 108)
- 1920-1921: Emil Post explora el problema de la interrupción de los sistemas de etiquetas, considerándolo como un candidato a la imposibilidad de resolver. (Problemas absolutamente irresolubles y proposiciones relativamente indecidibles: descripción de una anticipación, en Davis, 1965, págs. 340-433). Su insolubilidad no fue establecida hasta mucho más tarde, por Marvin Minsky (1967).
- 1928: Hilbert reformula su "Segundo problema" en el Congreso Internacional de Bolonia. (Reid págs. 188-189) Hodges afirma que planteó tres preguntas:
    - 1: ¿Las matemáticas estaban completas?
    - 2: ¿Las matemáticas fueron consistentes?
    - 3: ¿Fueron las matemáticas decidibles? (Hodges pág.91).
    - La tercera pregunta se conoce como Entscheidungsproblem (Problema de decisión). (Hodges p. 91, Penrose p. 34).
- 1930: Kurt Gödel anuncia una prueba como respuesta a las dos primeras preguntas de Hilbert de 1928 [cf. Reid p. 198]. "Al principio él [Hilbert] solo estaba enojado y frustrado, pero luego comenzó a tratar de abordar el problema de manera constructiva ... Gödel mismo sintió, y expresó la idea en su artículo, que su trabajo no contradecía el punto formalista de Hilbert (Reid pág. 199).

- 19 de abril de 1935: Alonzo Church pública "Un problema irresoluble de la teoría de números elementales", donde identifica lo que significa que una función sea efectivamente calculable. Tal función tendrá un algoritmo, y "... el hecho de que el algoritmo ha terminado se conoce efectivamente ..." (Davis, 1965, p. 100)
- 1937: El artículo de Alan Turing “On Computable Numbers With an Application to the Entscheidungs problem”, llega a la imprenta en enero de 1937 (reimpreso en Davis, 1965, p. 115). La demostración de Turing se aparta del cálculo mediante funciones recursivas e introduce la noción de cálculo por máquina. Stephen Kleene (1952) se refiere a esto como uno de los "primeros ejemplos de problemas de decisión que resultaron insolubles".
- 1952: "Martin Davis cree que es probable que haya utilizado por primera vez el término 'problema de detención' en una serie de conferencias que dio en el Laboratorio de Sistemas de Control de la Universidad de Illinois en 1952 (carta de Davis a Copeland, 12 de diciembre de 2001) ". (Nota 61 en Copeland (2004) pp. 40ff).

Hipótesis propuesta

El problema de la detención es un problema de decisión sobre las propiedades de los programas de computadora en un modelo de cálculo completo de Turing fijo , es decir, todos los programas que pueden escribirse en un lenguaje de programación dado que sea lo suficientemente general como para ser equivalente a una máquina de Turing. El problema es determinar, dado un programa y una entrada al programa, si el programa finalmente se detendrá cuando se ejecute con esa entrada. En este marco abstracto, no existen limitaciones de recursos sobre la cantidad de memoria o el tiempo requerido para la ejecución del programa; puede tardar arbitrariamente y utilizar una cantidad arbitraria de espacio de almacenamiento antes de detenerse. La pregunta es simplemente si el programa dado se detendrá alguna vez en una entrada en particular. Por ejemplo, en pseudocódigo , el programa ***(mientras (cierto) continuar)*** no se detiene; más bien, continúa para siempre en un bucle infinito . Por otro lado, el programa ***(imprimir "¡Hola, mundo!")*** se detiene.

Si bien decidir si estos programas se detienen es simple, los programas más complejos resultan problemáticos. Una forma de solucionar el problema puede ser ejecutar el programa durante una serie de pasos y comprobar si se detiene. Pero si el programa no se detiene, se desconoce si finalmente se detendrá o se ejecutará para siempre. Turing demostró que no existe un algoritmo que siempre decida correctamente si, para un programa y una entrada arbitrarios dados, el programa se detiene cuando se ejecuta con esa entrada. La esencia de la prueba de Turing es que se puede hacer que cualquier algoritmo de este tipo se contradiga y, por lo tanto, no sea correcto.

Mientras el problema mostrado no sea computable en la maquina de Turing, en la lógica combinatoria tampoco será computable, ya que el modelo de lógica combinatoria sólo funciona o se aplica en funciones que son computables y tienen una solución, caso contrario ocurre con el problema de Halting, ya que no se sabe cuando terminará la ejecución del problema.

Metodología

La metodología que se utilizará será la metodología ágil, debido a que en la metodología tradicional, un gestor de proyecto es el capitán del barco, lo que significa que todo le pertenece. Una de las diferencias notables en ambos enfoques de gestión de proyectos es el nivel de propiedad y responsabilidad que cada uno brinda a los miembros del equipo.

La metodología ágil tiene cuatro valores importantes:

- Mayor enfoque en individuos e interacciones que procesos y herramientas
- El software funcionando es más importante que una documentación extensa.
- La colaboración con el cliente es más importante que la negociación contractual.
- Responder al cambio en lugar de seguir ciegamente un plan.

Beneficios de la metodología ágil:

- Se establecen prioridades flexibles.
- Se empieza a entregar antes.
- Costes y plazos conocidos.

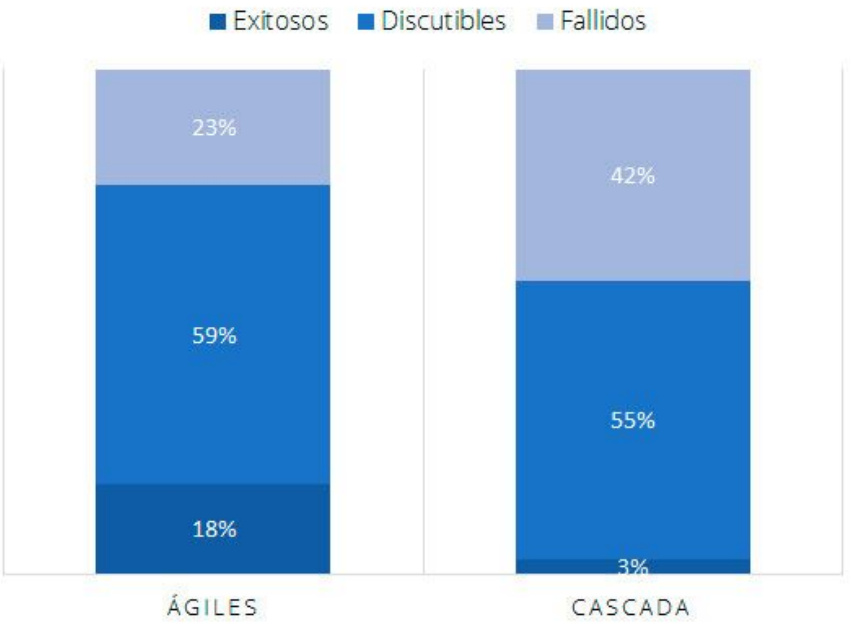
- Mejora la calidad final.
- Mayor transparencia.

“Agile” sigue un proceso iterativo en el que los proyectos se dividen en sprints de menor duración. A diferencia del enfoque tradicional, se gasta menos tiempo en la planificación y la priorización por adelantado, ya que el enfoque ágil es más flexible en cuanto a cambios respecto a los requerimientos iniciales.

En la metodología ágil, cada miembro del equipo comparte la propiedad del proyecto. Cada uno de ellos juega un papel activo para completar el sprint dentro del tiempo estimado. A diferencia del método tradicional, todos los involucrados en el proyecto pueden ver fácilmente el progreso desde el principio hasta el final.

La metodología ágil disfruta de una gran aceptación, se ha convertido en la primera opción para muchos gestores de proyectos porque pueden responder a las solicitudes requeridas a medida que validan cada iteración, esto les permite entregar un producto o servicio de alta calidad dentro del plazo establecido.

También nos hemos basado en la consultora The Standish Group, la cual, periódicamente publica el “CHAOS Report”, donde ilustran con datos el éxito o fracaso de los proyectos según la metodología utilizada, si comparamos ambas metodologías en un proyecto grande, vemos como según la metodología elegida tendremos mayor o menor probabilidad de éxito.



En conclusión, con una metodología ágil dividimos nuestro proyecto en partes más manejables, más pequeñas y lo vamos haciendo poco a poco, será más sencillo tener éxito.

REQUERIMIENTOS

Funcionales

- El problema a desarrollar debe ser computable.
- La lógica combinatoria sólo considera funciones lógicas en las que el resultado depende exclusivamente de las entradas.
- Se determinará si la lógica combinatoria es computable con la máquina de turing.
- El problema mostrado será sencillo de resolver.

No funcionales

- Permite calcular y almacenar tipos de datos.
- Permite modelos complejos a desarrollar.
- El problema planteado puede o no ser compatible con la máquina de turing.

Referencias

Wikipedia. (2020). Combinatory Logic. Recuperado el 06 de octubre del 2020 de: https://en.wikipedia.org/wiki/Combinatory_logic#Applications

Anónimo. (S.F). Combinatory Logic. Recuperado el 06 de octubre del 2020 de: https://wiki.haskell.org/Combinatory_logic

Wikipedia. (2020). Halting problem. Recuperado el 11 de Diciembre de 2020 de https://en.wikipedia.org/wiki/Halting_problem

Wikipedia. (2020). Problema de detención. Recuperado el 14 de Diciembre de https://es.qaz.wiki/wiki/Halting_problem

Escuela de negocios (feda). (2019). GESTIÓN ÁGIL VS GESTIÓN TRADICIONAL DE PROYECTOS ¿CÓMO ELEGIR?. Recuperado el 18 de diciembre de 2020 de https://www.escueladenegociosfeda.com/blog/50-la-huella-de-nuestros-docentes/471-gestion-agil-vs-gestion-tradicional-de-proyectos-como-elegir#:~:text=En%20la%20metodolog%C3%ADa%20%C3%A1gil%2C%20cada,comparte%20la%20propiedad%20del%20proyecto.&text=En%20el%20enfoque%20tradicional%2C%20cada,del%20tiempo%20y%20presupuesto%20estimados.