INSTITUTO TECNOLÓGICO DE IZTAPALAPA

COMPUTER SYSTEMS ENGINEER

Suggestion for the project development:

COMBINATORY LOGIC

Introduce:

ARREDONDO FLORES ALEXA KETZALI
BOHORQUEZ LOPEZ MIGUEL ANGEL
RODRIGUEZ GARCIA JESUS

Number of Control:

181080005
181080006
181080027

Internal Advisor:

PARRA HERNANDEZ ABIEL TOMAS

CIUDAD DE MÉXICO **NOVEMBER / 2020**

COMBINATORY LOGIC

Abstract

Combinatory Logic is a logical theory that is connected to many areas of logic, and has found applications in other disciplines, especially, in computer science and mathematics.

Combinatorial logic has a great importance in mathematics and computer science. In this project you will see some examples that use this logic, some of those examples are "Chaitin's construction".

With this project, it will be possible to better understand how combinatorial logic works, as a bit of its history.

Keywords: Combinatory Logic, mathematics, computer science.

Chronogram

Instituto Tecnologico de Iztapalapa
The chronogram of the project "Combinatory Logic"

Actividad	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Compilation of information about the investigation																
Construction of the summary and chronogram																
Brainstorm with all the information collected																
Construction of the project development																
Risk analysis																
Model description																
Investigation on the Halting problem																
Hypothesis and Proposal																
Investigation on the methodology to use																
Functional and Non-Functional Requirements																
Activity and Project Design																
Program to be carried out																
Investigation project delivery																

Risk Analysis

RISK	ANSWER
1.- Lack of knowledge and understanding of the topic.	The team will take the necessary time in research for a good understanding of the topic..
2.- Discrepancy and conflict between team members.	Execute group dynamics to strengthen teamwork and companionship.
3.- Lack of knowledge and understanding of software “JFLAP”.	The whole team must participate and try to work with the software for their understanding.
4.- Bad Time Estimates (Unrealistic Work Chronogram).	The team must commit to the project, the chronogram must be followed with the specified work time.
5.- Lack of activities of chronogram.	As the investigation progresses the relevant activities will be added.

Model description

Combinatory logic is a notation to eliminate the need for quantized variables in mathematical logic. It was introduced by Moses Schönfinkel and Haskell Curry, it is based on combiners that were introduced by Schönfinkel in 1920 with the idea of providing an analogous way of building functions and eliminating any mention of variables, particularly in predicate logic.

The goal of combinatorial logic was to better understand paradoxes and establish fundamental mathematical concepts on simpler and cleaner principles than existing mathematical frameworks, especially to better understand the concept of substitution.

Combinatorial logic is of great importance in the foundations of mathematics or computer science.

In mathematics, combinatorial logic was originally thought of as a "pre-logic" that would declare the role of quantized variables in logic, essentially eliminating them. In computer science, combinatorial logic is used as a simplified model of calculus, used in the theory of computability and the theory of proof. Despite its simplicity, combinatorial logic captures essential features of computation.

JUSTIFICATION

"Halting" Problem

In computability theory, the "halting problem" is the problem of determining, from a description of an arbitrary computer program and input, whether the program will finish running or continue running infinitely. Alan Turing showed in 1936 that there cannot be a general algorithm to solve the stop problem for all possible pairs of program input, that is, it is undecidable (not computable or non-recursive), in the sense that no Turing machine can solve it.

For any "f" program that can determine whether programs stop, a "pathological" program "g", called with some input, can pass its own source and input to "f" and then specifically do the opposite of what "F" predicts that "g" will do. There can be no person who handles this case. A key part of the proof is a mathematical definition of a computer and a program, known as a Turing machine; the problem of halting or "halting" is undecidable in Turing machines. It is one of the first cases of decision problems that have proven unsolvable. This test is significant for practical computing endeavors, as it defines a class of applications that no programming invention can perform perfectly.

A little history

The problem of detention is historically important because it was one of the first problems to be undecidable. (The Turing test was printed in May 1936, while Alonzo Church's test of the undecidability of a problem in lambda calculus had already been published in April 1936 [Church, 1936]). Subsequently, many other undecidable problems have been described:

Chronology

- 1900: David Hilbert poses his "23 questions" (now known as Hilbert problems) at the Second International Congress of Mathematicians in Paris.

"Of these, the second was to test the consistency of the 'Peano axioms' on which, as he had shown, the rigor of mathematics depended." (Hodges p. 83, Davis comment on Davis, 1965, p. 108)

- 1920-1921: Emil Post explores the problem of the interruption of label systems, considering it as a candidate for the impossibility of solving. (Absolutely unsolvable problems and relatively undecidable propositions: description of an anticipation, in Davis, 1965, pp. 340-433). His insolubility was not established until much later, by Marvin Minsky (1967).
- 1928: Hilbert reformulates his "Second Problem" at the International Congress in Bologna. (Reid pp. 188-189) Hodges claims that he posed three questions:
    - 1: Was the math complete?
    - 2: Was the math consistent?
    - 3: Was the math decidable? (Hodges p. 91).
    - The third question is known as Entscheidungsproblem (Decision problem). (Hodges p. 91, Penrose p. 34).
- 1930: Kurt Gödel announces a test in response to Hilbert's first two questions from 1928 [cf. Reid p. 198]. "At first he [Hilbert] was just angry and frustrated, but then he began to try to approach the problem constructively … Gödel himself felt, and expressed the idea in his article, that his work did not contradict Hilbert's formalist point. (Reid p. 199).
- April 19, 1935: Alonzo Church publishes "An Unsolvable Problem in Elementary Number Theory", where he identifies what it means for a function to be effectively calculable. Such a function will have an algorithm, and "… the fact that the algorithm has ended is effectively known …" (Davis, 1965, p. 100)
- 1937: Alan Turing's article "On Computable Numbers With an Application to the Entscheidungs problem" hits press in January 1937 (reprinted in Davis, 1965, p. 115). Turing's proof departs from recursive function calculus and introduces the notion of machine computation. Stephen Kleene (1952) refers to this as one of the "first examples of decision problems that proved insoluble."

- 1952: "Martin Davis believes that he probably first used the term 'stopping problem' in a series of lectures he gave at the University of Illinois Control Systems Laboratory in 1952 (Davis letter to Copeland, December 12, 2001) ". (Note 61 in Copeland (2004) pp. 40ff).

Proposed hypothesis

The stopping problem is a decision problem about the properties of computer programs in a fixed complete Turing calculus model, that is, all programs that can be written in a given programming language that is general enough to be equivalent to a Turing machine. The problem is determining, given a program and an input to the program, if the program will eventually stop when run with that input. In this abstract framework, there are no resource limitations on the amount of memory or the time required to run the program; it may take arbitrary time and use an arbitrary amount of storage space before stopping. The question is simply whether the given program will ever stop at a particular entry. For example, in pseudocode, the program **(while (true) continue)** does not stop; rather, it continues forever in an infinite loop. On the other hand, the program **(print "Hello world!")** Stops.

While deciding whether to stop these programs is simple, more complex programs are problematic. One way to fix the problem may be to run the program through a series of steps and see if it stops. But if the program doesn't stop, it is unknown whether it will eventually stop or run forever. Turing showed that there is no algorithm that always correctly decides whether, for a given arbitrary program and input, the program stops when executed with that input. The essence of the Turing test is that any such algorithm can be made to contradict itself and therefore not correct.

As long as the problem shown is not computable in the Turing machine, in combinatorial logic it will not be computable either, since the combinatorial logic model only works or is applied in functions that are computable and have a solution, otherwise it happens with the problem of Halting, since it is not known when the execution of the problem will end.

Methodology

The methodology that will be used will be the agile methodology, because in the traditional methodology, a project manager is the captain of the ship, which means that everything belongs to him. One of the notable differences in both project management approaches is the level of ownership and responsibility that each brings to team members.

The agile methodology has four important values:

- Greater focus on individuals and interactions than processes and tools
- Running software is more important than extensive documentation.
- Collaboration with the client is more important than contractual negotiation.
- Responding to change instead of blindly following a plan.

Benefits of the agile methodology:

- Flexible priorities are set.
- It begins to be delivered earlier.
- Known costs and deadlines.
- Improves the final quality.
- Greater transparency.

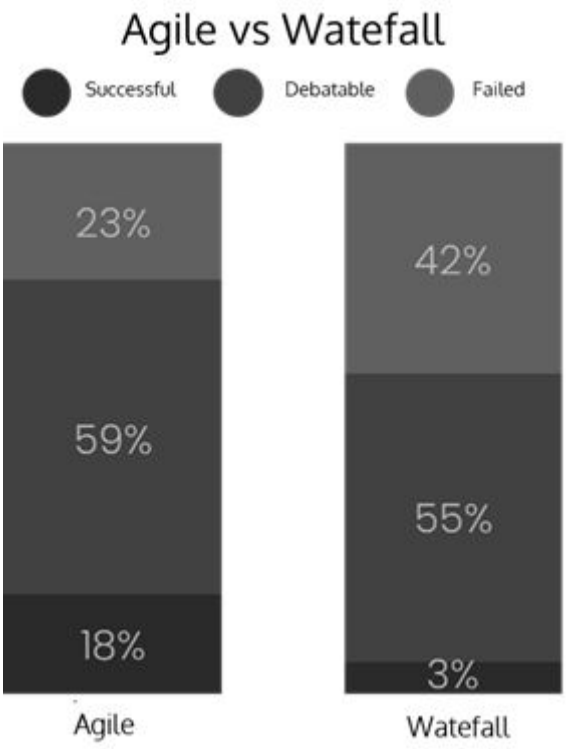
“Agile” follows an iterative process in which projects are divided into shorter sprints. Unlike the traditional approach, less time is spent planning and prioritizing in advance, as the agile approach is more flexible in terms of changes from initial requirements.

In the agile methodology, each team member shares ownership of the project. Each of them plays an active role in completing the sprint within the estimated time. Unlike the traditional method, everyone involved in the project can easily see the progress from the beginning to the end.

The agile methodology enjoys great acceptance, it has become the first choice for many project managers because they can respond to the required requests as they validate each iteration, this allows them to deliver a high quality product or service within the established deadline. .

We have also relied on the consulting firm The Standish Group, which periodically publishes the "CHAOS Report", where they illustrate with data the success or failure of projects according to the methodology used, if we compare both

methodologies in a large project, we see how according to the chosen methodology will have a greater or lesser probability of success.



In conclusion, with an agile methodology we divide our project into more manageable, smaller parts and we are doing it little by little, it will be easier to be successful.

REQUIREMENTS

Functional

- The problem to be developed must be computable.
- Combinatorial logic only considers logical functions in which the result depends exclusively on the inputs.
- It will be determined if the combinatorial logic is computable with the turing machine.
- The problem shown will be easy to solve.

Not functional

- Allows you to calculate and store data types.
- It allows complex models to develop.
- The problem posed may or may not be compatible with the Turing machine.

References

Wikipedia. (2020). Combinatory Logic. Retrieved on October 06, 2020 from: https://en.wikipedia.org/wiki/Combinatory_logic#Applications

Anonimous. (S.F). Combinatory Logic. Retrieved on October 06, 2020 from: https://wiki.haskell.org/Combinatory_logic

Wikipedia. (2020). Halting problem. Retrieved on December 11, 2020 from https://en.wikipedia.org/wiki/Halting_problem

Wikipedia. (2020). Problema de detención. Retrieved on December 14, 2020 from https://es.qaz.wiki/wiki/Halting_problem

Escuela de negocios (feda). (2019). GESTIÓN ÁGIL VS GESTIÓN TRADICIONAL DE PROYECTOS ¿CÓMO ELEGIR?. Retrieved on December 18, 2020 from 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.