UNIVERSIDAD NACIONAL HERMILIO VALDIZÁN - HUÁNUCO

FACULTAD DE CIENCIAS AGRARIAS

ESCUELA PROFESIONAL DE INGENIERÍA AGROINDUSTRIAL

"INFLUENCIA DE LA MANUFACTURA ASISTIDA POR COMPUTADORA (CAM) EN LA EFICIENCIA DEL PROCESO DE ELABORACIÓN DE NÉCTAR"

TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO AGROINDUSTRIAL

EJECUTOR : **Bach. Franklin Luis Salazar Cano**

ASESOR : **Mg. Gregorio Cisneros Santos**

HUÁNUCO – PERÚ

2018

DEDICATORIA

Dedico el presente trabajo de investigación científica a mi madre Teófila Maximina Cano Leandro y a mi padre Roger Salazar Suarez, quienes encaminaron mis pasos al éxito.

AGRADECIMIENTO

Agradezco en mi primer lugar a Dios, por darme la vida y salud para luchar en las duras etapas de la vida, asimismo agradezco a mis familiares y maestros por forjarme en el camino del conocimiento.

Por lo que mi compromiso es seguir la línea de la investigación científica, para colaborar con mi nación, la que me dió la educación.

RESUMEN

El presente investigación tuvo como objetivo determinar que el CAM (Manufactura Asistida por Computadora) a través del desarrollo de software influya en la eficiencia del proceso de elaboración de néctar, y como objetivos específicos se planteó desarrollar un software con estándares de calidad en codificación, para que contribuya en la eficiencia de la dosificación correcta de elaboración de néctar, desarrollar un software que influya eficientemente en el control del proceso de elaboración de néctar y determinar las características fisicoquímicas y sensoriales del néctar elaborado con la asistencia del software. Teniendo como problema fundamental que muchas empresas y microempresas no usan tecnologías en sus líneas de producción, la propuesta del software fue realizar un proceso más eficiente con información estructurada de alta calidad. Para determinar la aceptación del software se realizó el análisis estadístico con la prueba de Chi-cuadrado y la prueba T de muestras independientes, obteniendo como resultado que el software desarrollado si influye eficientemente en el proceso de elaboración de néctar, al realizar un proceso más eficiente utilizando información estructurada y de calidad, permitiendo reducir los tiempos en operaciones de cálculos para formulaciones y secuencias de las etapas del proceso, llevando un control estricto de toda la línea de producción; por lo que la mayoría de los encuestados respondieron que el software es importante para el desarrollo del proceso de producción, debido a la sistematización y automatización de la información, por lo que el 94% están dispuestos a adquirir el software. Se concluyó que el CAM a través del desarrollo de software si influye en la eficiencia en el proceso de elaboración de néctar, y se concluye de acuerdo a los objetivos específicos que el desarrollo del software con estándares de calidad en codificación contribuye eficientemente en la dosificación correcta de elaboración de néctar, y el desarrollo del software con los más altos estándares de calidad en diseño y codificación influye eficientemente en el control del proceso de elaboración de néctar.

Palabras clave: Automatización, desarrollo, programación, software.

ABSTRACT

The objective of the present investigation was to determine that CAM (Computer Aided Manufacturing) through software development influences the efficiency of the nectar elaboration process, and as specific objectives it was proposed to develop a software with coding quality standards, for that contributes in the efficiency of the correct dosage of nectar elaboration, to develop a software that influences efficiently in the control of the nectar elaboration process and to determine the physicochemical and sensorial characteristics of the nectar elaborated with the assistance of the software. Having as a fundamental problem that many companies and micro companies do not use technologies in their production lines, the software proposal was to carry out a more efficient process with structured information of high quality. To determine the acceptance of the software, the statistical analysis was performed with the chi-square test and the T test of independent samples, obtaining as a result that the software developed if it efficiently influences the nectar elaboration process, by performing a more efficient process using structured and quality information, allowing to reduce the time in operations of calculations for formulations and sequences of the stages of the process, taking strict control of the entire production line, also generating reports with exact data; so the majority of the respondents answered that the software is important for the development of the production process, due to the systematization and automation of the information, so that 94% are willing to acquire the software. It was concluded that the CAM (Computer Assisted Manufacturing) through the software development if it influences the efficiency in the nectar elaboration process, and it is concluded according to the specific objectives that the development of the software with quality standards in coding it contributes efficiently in the correct dosage of nectar elaboration, and the development of software with the highest quality standards in design and coding efficiently influences the control of the nectar elaboration process.

Keywords: Automation, development, programming, software.

ÍNDICE

DEDICATORIA 5
AGRADECIMIENTO 6
RESUMEN 7
ABSTRACT 8
I. INTRODUCCIÓN 13
II. MARCO TEÓRICO CONCEPTUAL 16
2.1. Fundamentación teórica 16
2.1.1. Tecnologías en la manufactura 16
2.1.2. Sistemas de manufactura 20
2.1.3. Manufactura integrada por computadora (CIM) 22
2.1.4. Manufactura asistida por computadora (CAM) 23
2.1.5. Diseño e ingeniería asistidos por computadora 25
2.1.6. Simulación por computadora de procesos y sistemas de manufactura 26
2.1.7. Tecnología de grupos 27
2.1.8. Tecnología de la automatización industrial 29
2.1.9. Evaluación de las inversiones en tecnología 30
2.1.10. Beneficios de las inversiones en tecnología 31
2.1.11. Software 34
2.1.12. Ingeniería del software 38
2.1.13. El proceso del software 40
2.1.14. Diseño en el contexto de la ingeniería de software 53
2.1.15. Arquitectura del software 55
2.1.16. Aseguramiento de la calidad del software 59
2.1.17. Programación orientado a objetos 62
2.1.18. Metodología RUP 67
2.1.19. Arduino 75
2.2. Antecedentes 81
2.3. Hipótesis 85
2.3.1. Hipótesis general 85
2.3.2. Hipótesis específicas 85

2.4. Variables....85
2.4.1. Variables independientes....85
2.4.2. Variables dependientes....85
2.4.3. Operacionalización de variables....86
III. MATERIALES Y MÉTODOS....87
3.1. Tipo y nivel de investigación....87
3.2. Lugar de ejecución....87
3.3. Población, muestra y unidad de análisis....87
3.4. Tratamiento en estudio....87
3.5. Prueba de hipótesis....88
3.5.1. Diseño de la investigación....88
3.5.2. Datos a registrar....91
3.5.3. Técnicas e instrumentos de recolección y procesamiento de datos de la información....91
3.6. Materiales y equipos....92
3.6.1. Materiales de laboratorio de programación....92
3.6.2. Materiales de escritorio....92
3.6.3. Equipos de programación....92
3.6.4. Insumos para elaboración del néctar....92
3.6.5. Materiales para la elaboración del néctar....93
3.6.6. Equipos para la elaboración del néctar....93
3.7. Conducción de la investigación....94
3.7.1. Etapa I.- Análisis de requisitos....95
3.7.2. Etapa II.- Diseño de la arquitectura....96
3.7.3. Etapa III.- Diseño y desarrollo del software para elaboración de néctar....97
3.7.4. Etapa IV.- Diseño y desarrollo del software para control del proceso....100
3.7.5. Etapa V.- Desarrollo del proceso de elaboración de néctar....109
3.7.6. Etapa VI.- Determinación de las características fisicoquímicas y sensoriales del néctar y análisis estadístico....116
IV. RESULTADOS....118

4.1. Etapa I.- Análisis de requisitos 118
a. Plan de iteración 118
b. Documento visión 123
c. Especificación de casos de uso 133
4.2. Etapa II.- Diseño de la arquitectura 136
4.3. Etapa III.- Diseño y desarrollo del software para elaboración de néctar 137
4.4. Etapa IV.- Diseño y desarrollo del software para control del proceso 138
4.5. Etapa V.- Desarrollo del proceso de elaboración de néctar 140
4.6. Etapa VI.- Determinación de las características fisicoquímicas y s ensoriales del néctar y análisis estadístico 142
4.6.1. Determinación de las características fisicoquímicas del néctar 142
4.6.2. Prueba Chi-cuadrado: aceptación del producto 143
4.6.3. Prueba Chi-cuadrado de asociación u homogeneidad: influencia de la programación en la elaboración de productos 146
4.6.4. Prueba T muestras independientes: Comparar el néctar elaborado con la utilización del software con néctar elaborado sin la utilización de software. 149
4.6.5. Gráficos de aceptación del software 157
V. DISCUSIÓN 162
5.1. Análisis de requisitos 162
5.2. Diseño de la arquitectura 162
5.3. Diseño y desarrollo del software para elaboración de néctar 162
5.4. Diseño y desarrollo del software para control del proceso 163
5.5. Desarrollo del proceso de elaboración de néctar 163
5.6. Determinación de las características fisicoquímicas y sensoriales del néctar y análisis estadístico 164
VI. CONCLUSIONES 165
VII. RECOMENDACIONES 167
VIII. LITERATURA CITADA 169
ANEXOS 172

ANEXO Nº 1: Diagrama de flujo del proceso de desarrollo del software. 172
ANEXO Nº 2: Documento glosario 173
ANEXO Nº 3: Matriz de consistencia 177
ANEXO Nº 4: Formatos de análisis estadístico 178

I. INTRODUCCIÓN

En la presente investigación se realizó el desarrollo de un software orientado al campo agroindustrial con una programación orientado a objetos (Full-OOP) utilizando para este medio como plataforma a Microsoft .Net Frameworks con el apoyo del programa Visual Studio (versión gratuita) con el lenguaje de Visual Basic, permitiéndonos desarrollar aplicativos e integración del software, estos aplicativos permiten monitorear parcialmente el determinado proceso, de tal manera que el usuario final tenga todo lo necesario a su disponibilidad mediante el uso del software.

El procedimiento de desarrollo para la creación del software fue en base a la NORMA IEEE 1058.1 y IEEE 15288-2004 normalizado para la ingeniería del software, donde nos brinda una serie instrucciones para la estructuración y codificación del software; seguidamente nos hemos apoyado en el principio de desarrollo de Haeber A. (1988) con su libro titulado “Formalización del proceso de desarrollo del software” donde nos brinda una amplia gama servicios para para la estructuración de los algoritmos, para el diseño de la arquitectura del software y los pasos correspondientes a seguir durante todo este proceso.

Después de haber realizado los análisis de los requerimientos para el software en base a los autores mencionados y otros se optó por desarrollar un software que cumpla los estándares de calidad, en cuanto a la arquitectura y al funcionamiento.

La presente investigación tuvo por objetivo determinar que el CAM (Manufactura Asistida por Computadora) a través del desarrollo de software influya en la eficiencia del proceso de elaboración de néctar; asimismo tuvimos como objetivo desarrollar un software con estándares de calidad en codificación, para que contribuya eficientemente en la dosificación correcta de elaboración de néctar, y también determinar las características fisicoquímicas y sensoriales del néctar elaborado con la asistencia del software.

El procedimiento de la investigación comenzó con la captura, elicitación, especificación y análisis de requisitos (ERS), siguiendo con el análisis y diseño general del software, y luego se comenzó con la estructuración de los

códigos y declaración de los algoritmos, realizado la programación se procedió con la integración y prueba del sistema, continuando con la instalación del software; realizado estos procedimientos se continuo con el análisis estadístico para evaluar la aceptación y la funcionalidad del software, donde se utilizó la prueba de Chi-cuadrado para evaluar la aceptación del software, y también se utilizó el estadístico de Chi-cuadrado con la prueba de asociación para evaluar la influencia de la programación estructurando información de calidad en la elaboración del néctar, y finalmente se utilizó la prueba t de muestras independientes para comparar la calidad del néctar elaborado bajo la asistencia del software y el néctar elaborado de manera convencional. Utilizando como materiales principalmente una computadora para la programación, y para la elaboración del néctar se utilizó los siguientes materiales: recipiente, jarras, coladores, tamiz, paletas, cucharas de medida, jaras de medida; y como equipos se utilizó pulpeadora, licuadora, balanza, refractómetro, pH-metro, termómetro; y como insumos para elaboración del néctar se utilizó, fruta cocona y mango, azúcar, estabilizante, conservante, reguladora de pH. El lugar donde se realizó la investigación fue ambientes del Centro de Innovación y Emprendimiento Agroindustrial, de la Escuela Académico Profesional de Ingeniería Agroindustrial de la Universidad Nacional Hermilio Valdizán en el departamento de Huánuco. Y el tiempo que duró la investigación fue en promedio de dos años.

Los objetivos planteados son:

General:

- Determinar que el CAM (Manufactura Asistida por Computadora) a través del desarrollo de software influye en la eficiencia del proceso de elaboración de néctar.

Específicos:

- Diseñar y desarrollar un software con estándares de calidad en codificación, para que contribuya en la eficiencia para la elaboración de néctar.
- Diseñar y desarrollar un software que influya eficientemente en el control del proceso de elaboración de néctar.
- Determinar las características fisicoquímicas y sensoriales del néctar elaborado con la asistencia del software.

II. MARCO TEÓRICO CONCEPTUAL

2.1. Fundamentación teórica

2.1.1. Tecnologías en la manufactura

Aquilano (2009) menciona que algunos de los adelantos tecnológicos registrados en decenios recientes han tenido un efecto generalizado en las compañías fabriles de muchas industrias. Dichos avances son el tema de esta sección y se pueden categorizar de dos maneras: los sistemas de hardware y los de software.

Las tecnologías de hardware por regla general han dado por resultado una mayor automatización de los procesos; desempeñan tareas que llevan mucho trabajo y que antes eran desempeñadas por humanos. Algunos ejemplos de estos tipos importantes de tecnologías de hardware son las máquinas herramienta controladas numéricamente, los centros de maquinado, los robots industriales, los sistemas automatizados para el manejo de materiales y los sistemas flexibles de producción. Todos ellos son instrumentos controlados por computadora que se pueden usar para la fabricación de productos. Las tecnologías basadas en software ayudan al diseño de los productos manufacturados y al análisis y la planeación de las actividades fabriles. Entre estas tecnologías se tiene el diseño asistido por computadora y la planeación automatizada de la manufactura y los sistemas de control. En las siguientes secciones se describe cada una de estas tecnologías con mayor detalle.

a. Sistemas de hardware

Las máquinas controladas numéricamente (CN) están compuestas por 1) una máquina herramienta típica usada para girar, perforar o troquelar diferentes tipos de piezas, y 2) una computadora que controla la secuencia de procesos que desarrolla la máquina. Las máquinas CN fueron adoptadas, por primera vez, por las empresas aeroespaciales de Estados Unidos en la década de 1960 y, desde entonces, han proliferado en muchas otras industrias. En los modelos más recientes, los ciclos del control de

retroalimentación determinan la posición de la herramienta de la máquina que está trabajando, comparan constantemente la situación actual con la programada y la corrigen cuando se necesita. Con frecuencia esto se llama un control de adaptación.

Figura 1. Uno de los cuatro grandes centros de maquinado que forman parte del sistema de producción flexible de la planta de Cincinnati Milacron, en Mt. Orab, Ohio.

Los *centros de maquinado* representan un grado mayor de automatización y complejidad en comparación con las máquinas de CN. Los centros de maquinado no sólo ofrecen control automático de una máquina, sino que también llevan muchas herramientas que pueden ser cambiadas automáticamente dependiendo de la herramienta requerida para cada operación. Además, una sola máquina podría estar equipada con un sistema de trenes de enlace de modo que la pieza terminada pueda ser descargada y la pieza sin terminar cargada, mientras la máquina está trabajando en una pieza.

Los *robots industriales* son empleados para sustituir a los trabajadores en muchas actividades manuales repetitivas y en tareas que son peligrosas, sucias o aburridas. Un robot es una máquina programable, que cumple con muchas funciones y que puede estar equipada con un actuador final. Algunos

ejemplos de actuadores finales serían unas pinzas para agarrar cosas o una herramienta como una llave, un cautín o un rociador de pintura. Ahora se han añadido capacidades a los robots para que haya una coordinación visual, de sensibilidad táctil y mano a mano. Además, a algunos modelos se les puede "enseñar" una secuencia de movimientos en un patrón tridimensional. Cuando un trabajador va llevando el extremo del brazo del robot por los movimientos requeridos, el robot registra este patrón en su memoria y lo repite cuando se le manda. Los sistemas robóticos más recientes pueden hacer inspecciones de control de calidad y después transferir, por vía de robots móviles, esas piezas a otros robots que se encuentran corriente abajo.

Los *sistemas para el manejo automatizado de materiales* (AMH, por sus siglas en inglés) mejoran la eficiencia del transporte, el almacenamiento y la recuperación de materiales. Algunos ejemplos serían las bandas computarizadas y los sistemas automatizados de almacenamiento y recuperación (AS/RS, por sus siglas en inglés), en cuyo caso las computadoras dirigen cargadores automáticos que levantan y colocan los artículos. Los sistemas de vehículos automatizados guiados (AGV, por sus siglas en inglés) usan cables incrustados en el piso para dirigir a vehículos sin conductor hacia distintos puntos de la planta. Algunos de los beneficios de los sistemas AMH son un movimiento más rápido de los materiales, inventarios y espacio de almacén más pequeños, menos daños a productos y mayor productividad laboral.

Estas piezas individuales automatizadas pueden ser combinadas para formar células de producción o incluso sistemas flexibles de producción completos (FMS, por sus siglas en inglés). Una célula de producción podría constar de un robot o de un centro de maquinado. El robot podría estar programado para que introduzca y saque, automáticamente, piezas del centro de maquinado, permitiendo con ello una operación no asistida. Un FMS es un sistema de producción totalmente automatizado que consta de centro de maquinado con carga y descarga automatizada de piezas, un sistema de vehículos automatizados guiados para mover las piezas entre máquinas y otros elementos automatizados que permiten la producción no asistida de

piezas. En un FMS, se usa un sistema de control general de computadora para manejar el sistema entero.

b. Sistemas de software

El diseño asistido por computadora (CAD, por sus siglas en inglés) es un enfoque para el diseño de productos y procesos que está basado en la potencia de la computadora. El CAD cubre varias tecnologías automatizadas, como las gráficas de computadora, para examinar las características visuales de un producto y la ingeniería asistida por computadora (CAE, por sus siglas en inglés) para evaluar sus características de ingeniería. Rubbermaid usó el CAD para afinar las dimensiones de sus carritos ToteWheels para cumplir con los requerimientos de las líneas aéreas para la facturación de equipaje. El CAD también incluye tecnología ligada al diseño de procesos de producción, llamada planeación de procesos asistidos por computadora (CAPP, por sus siglas en inglés), la cual se usa para diseñar en la computadora los programas de las partes que sirven de instrucciones para las máquinas herramienta controladas por computadora así como los usados para establecer la secuencia de las piezas que pasan por los centros de maquinado y por otros procesos (como lavado e inspección) necesarios para terminar una pieza. Estos programas se llaman planes del proceso. Los sistemas sofisticados de CAD pueden hacer pruebas en pantalla, reemplazando así las primeras fases de las pruebas y la modificación de prototipos.

El CAD se ha usado para diseñar de todo, desde chips de computadora hasta papas fritas. Frito-Lay, por ejemplo, usó el CAD para diseñar sus papas fritas acanaladas O'Grady de doble grosor. El problema del diseño de estas papas es que, si no se cortan debidamente, se pueden quemar por fuera y quedar medio crudas por dentro, quedar demasiado crujientes (y romperse cuando se meten en la bolsa) o exhibir otras características que las hacen inservibles para, por ejemplo, mojar en una salsa de guacamole. No obstante, gracias al CAD, fue posible determinar matemáticamente el ángulo y el número adecuado de papas acanaladas y el modelo O'Grady pasó su prueba de presión en la infame "aplastadora" de Frito-Lay y llegó a los anaqueles de

la tienda. Sin embargo, a pesar de algunos leales consumidores, O'Grady's ha sido descontinuada en razón de sus escasas ventas.

Ahora el CAD se está usando para diseñar trajes de baño a la medida. Las medidas del usuario son alimentadas al programa del CAD, así como el modelo de traje que desea. Trabajando con el cliente, el diseñador modifica el diseño del traje que aparece en la pantalla de la computadora en una imagen de forma humana. Una vez decidido el diseño, la computadora imprime un patrón y el traje es cortado y cosido de inmediato.

Los s*istemas automatizados de planeación y control de la producción* (MP&CS, por sus siglas en inglés) son simples sistemas de información computarizada que ayudan a planear, programar y vigilar una operación fabril. Éstos obtienen constantemente información del taller de la planta acerca de la situación del trabajo, las llegadas de materiales, etc., y libran las órdenes de compras y de producción. Los sistemas complejos de producción y control de la planeación incluyen el procesamiento de las órdenes que ingresan, el control del taller de la planta, las adquisiciones y la contabilidad de costos.

2.1.2. Sistemas de manufactura

Kalpakjian y Schmid (2008) Menciona que el término sistema se deriva de la palabra griega *systema*, que significa "combinar". El sistema ha llegado a significar el arreglo de entidades físicas con parámetros de interacción identificables y cuantificables. Como ya se vio en diversos capítulos, la manufactura conlleva una gran cantidad de actividades interdependientes que constan de distintas entidades, de ahí que pueda tratarse como un sistema.

La manufactura es un sistema complejo porque consta de muy diversos elementos físicos y humanos, algunos de los cuales son difíciles de predecir y controlar. Entre estas dificultades figuran factores como el suministro y costo de materias primas, cambios de mercados nacionales y globales, el impacto de tecnologías en constante desarrollo, así como el comportamiento y desempeño humanos. En términos ideales, un sistema de manufactura debe representarse mediante modelos matemáticos y físicos que muestren la naturaleza y el grado de interdependencia de todas las variables comprendidas. De esta manera, se pueden analizar los efectos de un cambio

o perturbación ocurrido en cualquier parte del sistema y realizarse los ajustes necesarios.

Por ejemplo, el suministro de una materia prima en particular puede reducirse de manera significativa debido a, digamos, demandas globales, guerras, huelgas o geopolítica. Como consecuencia, el costo de la materia prima aumentaría y quizá tuvieran que buscarse y elegirse materiales alternativos. Esta selección debe realizarse después de considerar con cuidado diversos factores, ya que dicho cambio puede tener efectos adversos sobre la calidad del producto, la capacidad de producción y los costos de manufactura. Por ejemplo, es posible que no sea tan fácil dar forma, maquinar o soldar el material seleccionado, o que la integridad del producto sufra daños durante su procesamiento.

En un mercado en constante cambio, la demanda de un producto puede fluctuar aleatoria y rápidamente por diversas razones. Como ejemplos, considérese la disminución de dimensiones de los automóviles durante la década de 1980 en respuesta a la escasez de combustible y, en contraste, la reciente popularidad de los vehículos deportivos y el interés actual en los híbridos de gas y eléctricos. Así, el sistema debe tener la capacidad de producir el producto modificado en un tiempo de entrega relativamente corto y de reducir al mínimo los gastos en maquinaria y herramientas.

Tal vez sea difícil analizar y modelar un sistema tan complejo debido a la falta de información amplia o confiable sobre todas las variables comprendidas. Tampoco es fácil predecir ni controlar de manera correcta algunas de estas variables. Por ejemplo: *(a)* las características de las máquinas herramienta, su desempeño y su respuesta a alteraciones externas aleatorias no pueden modelarse con precisión; *(b)* es complicado predecir con precisión los costos de la materia prima, y (c) es difícil modelar el comportamiento y desempeño humanos. A pesar de las dificultades, se han realizado muchos avances en el modelado y la simulación de los sistemas de manufactura.

2.1.3. Manufactura integrada por computadora (CIM)

Aquilano (2009) indica todas estas tecnologías automatizadas quedan reunidas en la manufactura integrada por computadora (CIM, por sus siglas en inglés). La CIM es una versión automatizada del proceso de producción, en la cual las tres funciones básicas de ésta (diseño del proceso y el producto, la planeación y el control, y el proceso de producción mismo) son reemplazadas por las tecnologías automatizadas que se acaban de describir. Es más, los mecanismos tradicionales de integración de la comunicación oral y escrita son reemplazados por la tecnología de cómputo. Esta producción tan integrada y automatizada también se conoce con el nombre de automatización total de la fábrica y fábrica del futuro.

Todas las tecnologías de la CIM quedan reunidas mediante una red y una base de datos integrados. Por ejemplo, la integración de datos permite ligar los sistemas de CAD a la manufactura asistida por computadora (CAM), la cual está compuesta por programas de control numérico de las piezas y el sistema de planeación; además, es posible ligar el sistema de planeación y control de la producción con los sistemas automatizados de manejo de materiales a efecto de que se genere la lista de piezas que se tomarán. Así pues, en un sistema plenamente integrado, las áreas de diseño, prueba, fabricación, montaje, inspección y manejo de materiales no sólo están automatizadas sino también están todas integradas entre sí y con la función de planeación y programación de la producción.

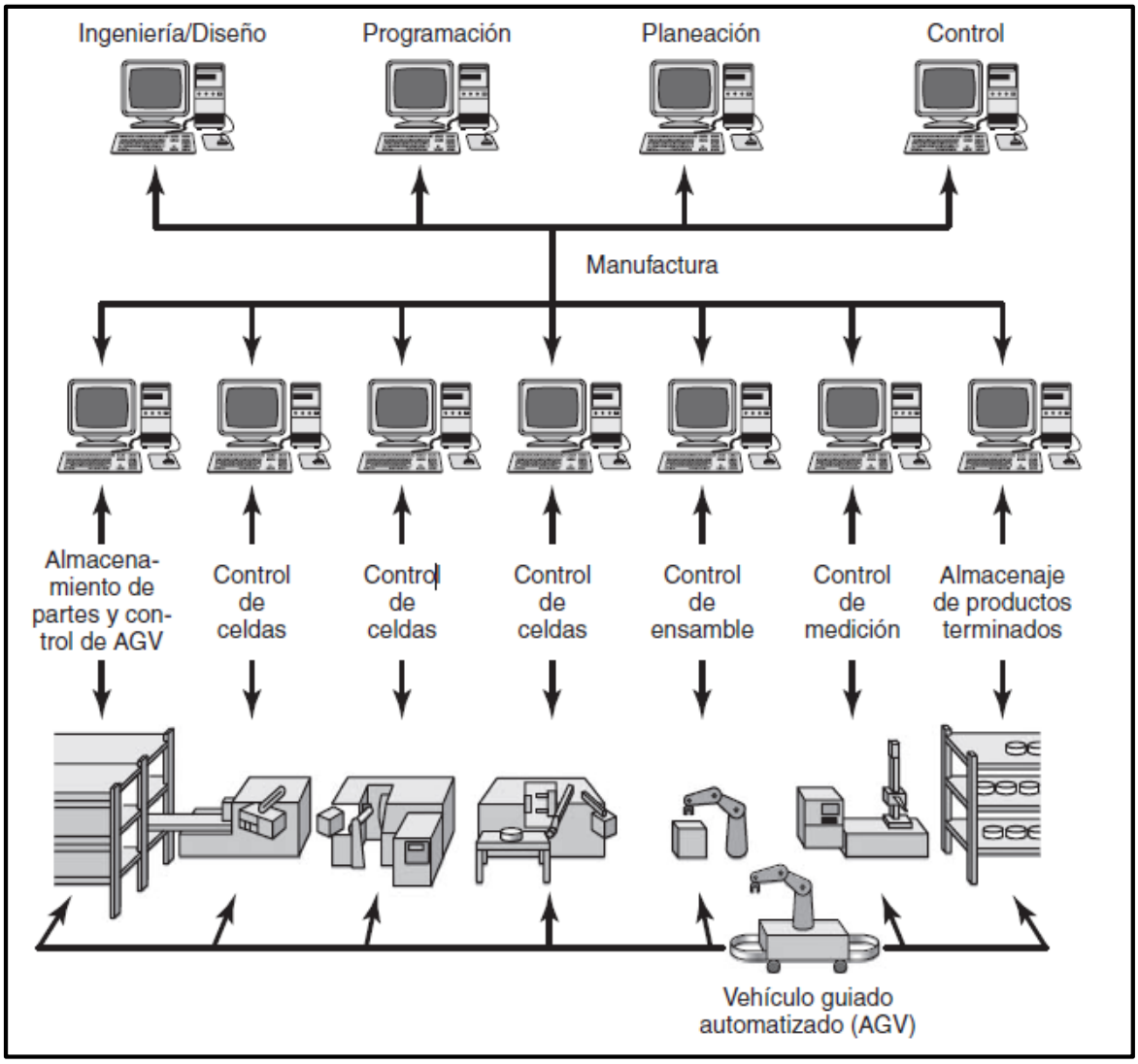

Figura 2. Esquema de un sistema de manufactura integrado por computadora.

2.1.4. Manufactura asistida por computadora (CAM)

Kalpakjian y Schmid (2008) menciona que la manufactura asistida por computadora (CAM, por sus siglas en inglés) comprende el uso de computadoras para auxiliar en todas las fases de manufactura de un producto. Debido a los beneficios conjuntos, a menudo el diseño asistido por computadora y la manufactura asistida por computadora se combinan en sistemas CAD/CAM. Esta combinación permite transferir información de la etapa de diseño a la etapa de planeación de manufactura sin necesidad de introducir manualmente los datos sobre la geometría de la parte otra vez. La

CAM almacena y procesa la base de datos desarrollada durante el CAD con los datos e instrucciones necesarios para operar y controlar maquinaria de producción, equipo de manejo de materiales y ensayos e inspección automatizados para alcanzar la calidad de los productos. Los sistemas CAD/CAM también cuentan con la capacidad de codificar y clasificar partes en grupos que tienen formas similares utilizando codificación alfanumérica.
Un rasgo importante de CAD/CAM en las operaciones de maquinado es su capacidad para describir la trayectoria de las herramientas. Las instrucciones (programas) se generan en la computadora y el programador las puede modificar para optimizar la trayectoria de las herramientas. El ingeniero o técnico puede desplegar y verificar en forma visual la trayectoria de las herramientas en caso de posibles colisiones con las prensas de sujeción, soportes fijos u otras interferencias.

Al estandarizar el desarrollo de los productos y reducir el esfuerzo del diseño, pruebas y trabajo de los prototipos, CAD/CAM ha hecho posible reducir de modo significativo los costos de manufactura y ha mejorado la productividad. Por ejemplo, el avión de pasajeros Boeing 777 de dos motores se diseñó totalmente por computadora (diseño sin documentos: paperless design), con 2000 estaciones de trabajo conectadas a ocho computadoras. El plano se construyó en forma directa del software de CAD/CAM desarrollado (un sistema CATIA mejorado) y no se construyeron prototipos ni maquetas, como se requirió para modelos anteriores. El costo de este desarrollo fue de $6 mil millones de dólares.

Algunas aplicaciones comunes de CAD/CAM son las siguientes:

- Programación de control numérico y robots industriales.
- Diseño de matrices o dados y moldes para fundición, por ejemplo, en los que se programan anticipadamente las tolerancias por contracción.
- Matrices o dados para operaciones de trabajo mecánico, como matrices complejas para formado de láminas y matrices progresivas para estampado.
- Diseño de herramental y soportes fijos, y electrodos EDM.

- Control de calidad e inspección, como las máquinas de medición por coordenadas programadas en una estación de trabajo de CAD/CAM.
- Planeación y programación de proceso.
- Distribución de planta.

2.1.5. Diseño e ingeniería asistidos por computadora

Kalpakjian y Schmid (2008) menciona que el diseño asistido por computadora (CAD, por sus siglas en inglés) comprende el uso de computadoras para crear planos de diseño y modelos de productos. Por lo común, el diseño asistido por computadora se asocia con gráficos interactivos por computadora, conocidos como sistema CAD. Los sistemas de diseño asistido por computadora son herramientas poderosas y se utilizan en el diseño y modelado geométrico de componentes y productos.

Los planos se generan en estaciones de trabajo y el diseño se muestra de modo continuo en el monitor con diferentes colores para los distintos componentes. El diseñador puede conceptualizar con facilidad la parte a diseñar en las pantallas de gráficos y considerar propuestas alternativas o modificar con rapidez un diseño específico que satisfaga requisitos particulares. Con el uso de software poderoso como CATIA (siglas en inglés de aplicaciones interactivas tridimensionales asistidas por computadora), el diseño puede someterse al análisis de ingeniería e identificar problemas potenciales, como exceso de carga, deflexión o interferencia en superficies de contacto durante el ensamble. También se almacena información (listas de materiales, especificaciones e instrucciones de manufactura) en la base de datos de CAD. Utilizando esta información, el diseñador puede analizar la economía de manufactura de diseños alternativos.

La ingeniería asistida por computadora (CAE, por sus siglas en inglés) permite que diversas aplicaciones compartan la información en la base de datos. Estas aplicaciones incluyen (a) el análisis de elementos finitos de esfuerzos, deformaciones, deflexiones y distribución de temperatura en estructuras y miembros de soporte de carga; (b) la generación, el almacenaje

y la recuperación de datos de NC, y (c) el diseño de circuitos integrados y de diversos dispositivos electrónicos.

2.1.6. Simulación por computadora de procesos y sistemas de manufactura

Kalpakjian y Schmid (2008) menciona con la creciente sofisticación del equipo y software para computadoras, la simulación por computadora de los procesos y sistemas de manufactura ha avanzado con rapidez. La simulación tiene dos formas básicas:

- Es un modelo de operación específica que tiene el propósito de determinar la viabilidad de un proceso u optimizar o mejorar su desempeño.
- Modela múltiples procesos y sus interacciones para ayudar a los planeadores de procesos y diseñadores de plantas en la disposición de la maquinaria e instalaciones.

Se han modelado procesos individuales utilizando diversos esquemas matemáticos. De manera creciente, se ha aplicado el análisis de elementos finitos en los paquetes (simulación de procesos) disponibles comercialmente y son poco costosos. Los problemas comunes enfocados son la viabilidad de los procesos (como la formabilidad de la lámina metálica en cierta matriz o dado) y la optimización del proceso (como flujo del material en el forjado de una matriz determinada a fin de identificar defectos potenciales, o diseños de moldes en fundición para eliminar puntos calientes, promover un enfriamiento uniforme y minimizar los defectos).

La simulación de todo un sistema de manufactura que comprende procesos y equipos múltiples ayuda a los ingenieros de planta a organizar la maquinaria e identificar elementos críticos de la misma. Además, dichos modelos pueden ayudar a los ingenieros de manufactura con la programación y las rutas (mediante simulación de eventos discretos). Para estas simulaciones se utilizan paquetes de software disponibles comercialmente, pero también se pueden desarrollar programas de software escritos para una compañía específica.

2.1.7. Tecnología de grupos

Kalpakjian y Schmid (2008) menciona que muchas partes de los productos tienen ciertas similitudes en su forma y en su método de manufactura. La tecnología de grupos (GT, por sus siglas en inglés) es un concepto que busca aprovechar las similitudes de diseño y procesamiento entre las partes a producir. El término “tecnología de grupos” se utilizó por primera vez en 1959, pero no fue sino hasta que aumentó el uso de computadoras interactivas en la década de 1970 que esta tecnología se desarrolló de manera significativa.

La similitud en las características de partes similares sugiere que se pueden obtener beneficios mediante la clasificación y codificación de estas partes en familias. Al desensamblar cada producto en sus componentes individuales e identificar las partes similares, una compañía descubrió que 90% de las 3000 partes fabricadas por una compañía entraban en sólo cinco familias principales.

Por ejemplo, una bomba puede dividirse en componentes básicos como motor, carcasa, eje, bridas y sellos. A pesar de la variedad de bombas manufacturadas, cada uno de estos componentes es el mismo en términos de diseño y características de manufactura. Por consiguiente, todos los ejes se pueden colocar en una misma familia y así sucesivamente. Además, se ha cuestionado por qué un producto en particular debe tener tantos tamaños diferentes de sujetadores.

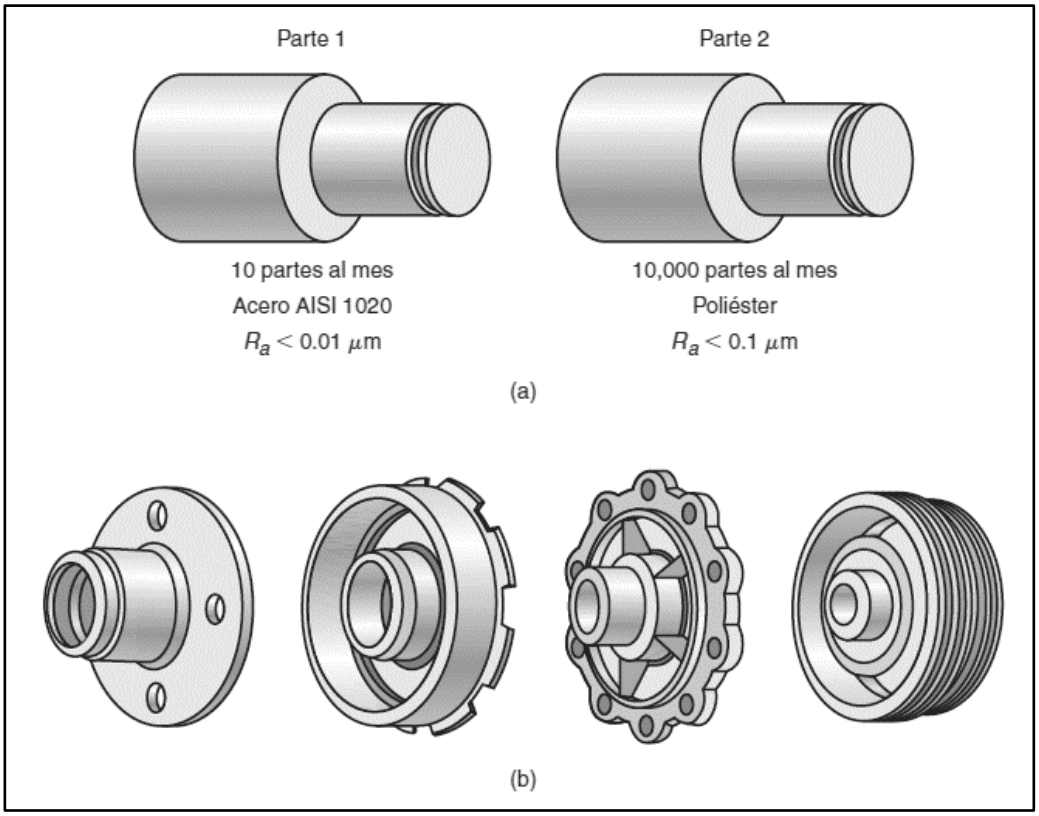

Figura 3. Agrupación de partes de acuerdo con (a) sus similitudes geométricas, y (b) atributos de manufactura.

La tecnología de grupos se volvió muy atractiva debido a la variedad cada vez mayor de productos disponibles para el consumidor, que a menudo se producen en lotes. Dado que hoy en día casi 75% de la manufactura es producción en lote, se ha vuelto importante mejorar la eficiencia de dicha producción.

En la figura anterior se muestra el flujo tradicional de productos en la manufactura en lotes. Obsérvese que se ordenan máquinas del mismo tipo en grupos; es decir, grupos de tornos, fresadoras, taladradoras y rectificadoras. En dicha distribución (conocida como distribución funcional), por lo general existe considerable movimiento aleatorio, como lo muestran las flechas que indican movimiento de materiales y partes. Dicho arreglo no es eficiente, porque desperdicia tiempo y esfuerzos. Una línea de flujo de productos más eficiente para aprovechar la tecnología de grupos es el arreglo o distribución de grupos.

2.1.8. Tecnología de la automatización industrial

Descripción tecnológica del automatismo

Gaither y Frazier (2006) indica que la descripción tecnológica del automatismo es el conjunto de elementos físicos que lo forman. En concreto, estos elementos son los sensores, los actuadores y el sistema de control. Por otra parte está la descripción funcional, que se refiere a las características de funcionamiento del sistema automatizado.

Los sensores son los elementos que permiten obtener información de lo que sucede en el proceso. Se pueden distinguir dos tipos de sensores, según la información que proporcionan:

Tipos de automatización

Gaither y Frazier (2006) indica que los **La automatización fija** se utiliza cuando el volumen de producción es muy alto, y por tanto se puede justificar económicamente el alto costo del diseño de equipo especializado para procesar el producto, con un rendimiento alto y tasas de producción elevadas. Además de esto, otro inconveniente de la automatización fija es su ciclo de vida que va de acuerdo a la vigencia del producto en el mercado.

La justificación económica para la automatización fija se encuentra en productos con grandes índices de demanda y volumen.

La **automatización programable** se emplea cuando el volumen de producción es relativamente bajo y hay una diversidad de producción a obtener. En este caso el equipo de producción es diseñado para adaptarse a la variaciones de configuración del producto; ésta adaptación se realiza por medio de un programa (Software).

- Fuerte inversión en equipo general
- Índices bajos de producción para la automatización fija
- Flexibilidad para lidiar con cambios en la configuración del producto
- Conveniente para la producción en montones

La **automatización flexible** es más adecuada para un rango de producción medio. Estos sistemas flexibles poseen características de la automatización fija y de la automatización programada. Los sistemas flexibles suelen estar constituidos por una serie de estaciones de trabajo interconectadas entre sí por sistemas de almacenamiento y manipulación de materiales, controlados en su conjunto por una computadora.

Proveedores de equipos de automatización

- Fuerte inversión para equipo de ingeniería
- Producción continúa de mezclas variables de productos
- Índices de producción media
- Flexibilidad para lidiar con las variaciones en diseño del producto

Las características esenciales que distinguen la automatización flexible de la programable son:

- Capacidad para cambiar partes del programa sin perder tiempo de producción y;
- Capacidad para cambiar sobre algo establecido físicamente asimismo sin perder tiempo de producción.

2.1.9. Evaluación de las inversiones en tecnología

Gaither y Frazier (2006) indica que las tecnologías modernas como los sistemas flexibles de producción o los sistemas computarizados de procesamiento de órdenes representan cuantiosas inversiones de capital. Por lo mismo, antes de adquirir una tecnología, la empresa debe evaluar con sumo cuidado los beneficios financieros y estratégicos que obtendrá de ella. La evaluación de estas inversiones es especialmente difícil porque el propósito de adquirir nuevas tecnologías no es tan sólo reducir los costos del trabajo, sino también elevar la calidad y la variedad de los productos, acortar los tiempos de espera de la producción y aumentar la flexibilidad de una operación. Algunos de estos beneficios son intangibles en relación con la reducción de costos del trabajo, por lo que resulta muy difícil justificarlos. Es más, el veloz cambio de la tecnología hace que el equipamiento nuevo quede

obsoleto en pocos años, complicando aún más la evaluación de los costos y los beneficios.

Sin embargo, jamás suponga que las nuevas tecnologías de automatización siempre tienen costos efectivos. Incluso cuando no exista incertidumbre con respecto a los beneficios de la automatización, podría no valer la pena adoptarla. Por ejemplo, muchos analistas supusieron que los sistemas integrados CAD/CAM serían la respuesta a todos los problemas de producción. Sin embargo, una serie de compañías que invirtieron en estos sistemas perdieron dinero al hacerlo. La idea era sacar a un montón de trabajadores calificados del proceso de preparar las máquinas para productos nuevos o rediseñados y acelerar el proceso. No obstante, podría tomar menos tiempo fabricar piezas complejas en volúmenes pequeños que programar la máquina fresadora, y el tiempo del programador es más caro que el del operador de la fresa. Además, toda la experiencia y el conocimiento especializado que un operador fresador ha adquirido a lo largo de los años no siempre se podrá transferir con facilidad a un programa de computadora. El software integrado CAD/CAM ha alcanzado niveles de calidad y eficiencia de costos bastantes para justificar que se use, de forma rutinaria, hasta en contextos fabriles que manejan gran variedad y poco volumen.

2.1.10. Beneficios de las inversiones en tecnología

Aquilano (2009) menciona que los beneficios típicos de adoptar nuevas tecnologías de producción son tangibles e intangibles. Los tangibles se pueden emplear en las formas tradicionales del análisis financiero, como el flujo de efectivo descontado, para tomar decisiones de inversión sólidas. Los beneficios específicos quedarían resumidos así:

Reducción de costos

Costos laborales: reemplazar a las personas con robots o permitir que un número menor de trabajadores maneje equipamiento semiautomático.

Costos de materiales: usar con más eficiencia los materiales existentes o permitir el uso de materiales de gran tolerancia.

Costos de inventarios: equipamiento que se puede cambiar rápidamente, lo cual permite la administración de un inventario JIT.

Costos de calidad: inspección automatizada y menor variación en los productos elaborados.

Costos de mantenimiento: equipamiento que se ajusta solo.

Otros beneficios

Mayor variedad de productos: economías de alcance en razón de sistemas flexibles de producción.

Mejores características de los productos: capacidad para hacer cosas que no se podrían hacer a mano (v. gr., microprocesadores).

Ciclos más cortos de tiempo: mayor velocidad para preparar las **máquinas o cambiarlas.**

Mayor producción de productos.

Riesgos de adoptar las nuevas tecnologías

Si bien la adquisición de nuevas tecnologías entraña muchos beneficios, ésta también implica varios tipos de riesgo. Antes de adoptar las tecnologías es preciso evaluar estos riesgos y sopesarlos con los beneficios. A continuación se describen algunos de los riesgos.

a. Riesgos tecnológicos

Una compañía que es de las primeras en adoptar una nueva tecnología tiene el beneficio de llevarle la delantera a la competencia, pero también corre el riesgo de adquirir una tecnología que no ha sido probada y que podría traer problemas que alteran las operaciones de la empresa. También está el riesgo de la obsolescencia, sobre todo en el caso de las tecnologías electrónicas

donde el cambio es veloz y el costo fijo de las nuevas tecnologías o el costo de las actualizaciones son elevados. Asimismo, las tecnologías alternativas podrían representar costos más efectivos en el futuro, anulando los beneficios de la tecnología de hoy.

b. Riesgos para las operaciones

También puede haber riesgos cuando una empresa aplica una nueva tecnología a sus operaciones. La instalación de una nueva tecnología por lo general produce alteraciones sustantivas, cuando menos al corto plazo, en la forma de reorganizar toda la planta, la recapacitación del personal, etc. Otros riesgos se deben a las demoras y los errores que se introducen en el proceso de producción y las demandas inciertas y repentinas impuestas a diversos recursos.

c. Riesgos para la organización

Las empresas podrían carecer de la cultura organizacional y el compromiso de la alta gerencia necesario para absorber las alteraciones y las incertidumbres de corto plazo asociadas a la adopción de una nueva tecnología. En estas organizaciones existe el riesgo de que los empleados o los administradores de la empresa puedan abandonar rápidamente la tecnología cuando se presentan fallas a corto plazo o evitarán cambios mayores automatizando simplemente el viejo proceso ineficiente de la empresa y, por lo tanto, no se obtendrán los beneficios de la nueva tecnología.

d. Riesgos para el ambiente o el mercado

En muchos casos, una empresa podría invertir en una tecnología concreta, tan sólo para encontrar pocos años después que los cambios en algunos factores del ambiente o del mercado hacen que su inversión no valga nada. Por ejemplo, en cuestiones ambientales, las compañías automovilísticas no han querido invertir en tecnología para fabricar automóviles eléctricos porque no están seguras de cuáles serán las normas de los gobiernos estatales y el federal que rijan las emisiones en el futuro ni del potencial para disminuir las emisiones de los automóviles de gasolina y el

potencial para mejoras sustantivas en la tecnología de las baterías. Algunos ejemplos típicos de los riesgos del mercado son las fluctuaciones de los tipos de cambio de las divisas y de las tasas de interés.

2.1.11. Software

Sommerville (2005) menciona que en la actualidad, el software tiene un papel dual. Es un producto y al mismo tiempo es el vehículo para entregar un producto. En su forma de producto, brinda el potencial de cómputo incorporado en el hardware de cómputo o, con más amplitud, en una red de computadoras a las que se accede por medio de un hardware local. Ya sea que resida en un teléfono móvil u opere en el interior de una computadora central, el software es un transformador de información produce, administra, adquiere, modifica, despliega o transmite información que puede ser tan simple como un solo bit o tan compleja como una presentación con multimedios generada a partir de datos obtenidos de decenas de fuentes independientes. Como vehículo utilizado para distribuir el producto, el software actúa como la base para el control de la computadora (sistemas operativos), para la comunicación de información (redes) y para la creación y control de otros programas (herramientas y ambientes de software).

En la actualidad, la enorme industria del software se ha convertido en un factor dominante en las economías del mundo industrializado. Equipos de especialistas de software, cada uno centrado en una parte de la tecnología que se requiere para llegar a una aplicación compleja, han reemplazado al programador solitario de los primeros tiempos. A pesar de ello, las preguntas que se hacía aquel programador son las mismas que surgen cuando se construyen sistemas modernos basados en computadora:

- ¿Por qué se requiere tanto tiempo para terminar el software?
- ¿Por qué son tan altos los costos de desarrollo?
- ¿Por qué no podemos detectar todos los errores antes de entregar el software a nuestros clientes?
- ¿Por qué dedicamos tanto tiempo y esfuerzo a mantener los programas existentes?

- ¿Por qué seguimos con dificultades para medir el avance mientras se desarrolla y mantiene el software?

Éstas y muchas otras preguntas, denotan la preocupación sobre el software y la manera en que se desarrolla, preocupación que ha llevado a la adopción de la práctica de la ingeniería del software.

a. Definición de software

1. Son instrucciones (programas de cómputo) que cuando se ejecutan proporcionan las características, función y desempeño buscados.
2. Estructuras de datos que permiten que los programas manipulen en forma adecuada la información y
3. Información descriptiva tanto en papel como en formas virtuales que describen la operación y uso de los programas.

b. Dominios de aplicación del software

Actualmente, hay siete grandes categorías de software de computadora que plantean retos continuos a los ingenieros de software:

Software de sistemas: conjunto de programas escritos para dar servicio a otros programas. Determinado software de sistemas (por ejemplo, compiladores, editores y herramientas para administrar archivos) procesa estructuras de información complejas pero deterministas. Otras aplicaciones de sistemas (por ejemplo, componentes de sistemas operativos, manejadores, software de redes, procesadores de telecomunicaciones) procesan sobre todo datos indeterminados. En cualquier caso, el área de software de sistemas se caracteriza por: gran interacción con el hardware de la computadora, uso intensivo por parte de usuarios múltiples, operación concurrente que requiere la secuenciación, recursos compartidos y administración de un proceso sofisticado, estructuras complejas de datos e interfaces externas múltiples.

Software de aplicación: programas aislados que resuelven una necesidad específica de negocios. Las aplicaciones en esta área procesan datos comerciales o técnicos en una forma que facilita las operaciones de negocios o la toma de decisiones administrativas o técnicas. Además de las aplicaciones convencionales de procesamiento de datos, el software de aplicación se usa para controlar funciones de negocios en tiempo real (por ejemplo, procesamiento de transacciones en punto de venta, control de procesos de manufactura en tiempo real).

Software de ingeniería y ciencias: se ha caracterizado por algoritmos "devoradores de números". Las aplicaciones van de la astronomía a la vulcanología, del análisis de tensiones en automóviles a la dinámica orbital del transbordador espacial, y de la biología molecular a la manufactura automatizada. Sin embargo, las aplicaciones modernas dentro del área de la ingeniería y las ciencias están abandonando los algoritmos numéricos convencionales. El diseño asistido por computadora, la simulación de sistemas y otras aplicaciones interactivas, han comenzado a hacerse en tiempo real e incluso han tomado características del software de sistemas.

Software incrustado: reside dentro de un producto o sistema y se usa para implementar y controlar características y funciones para el usuario final y para el sistema en sí. El software incrustado ejecuta funciones limitadas y particulares (por ejemplo, control del tablero de un horno de microondas) o provee una capacidad significativa de funcionamiento y control (funciones digitales en un automóvil, como el control del combustible, del tablero de control y de los sistemas de frenado).

Software de línea de productos: es diseñado para proporcionar una capacidad específica para uso de muchos consumidores diferentes. El software de línea de productos se centra en algún mercado limitado y particular (por ejemplo, control del inventario de productos) o se dirige a mercados masivos de consumidores (procesamiento de textos, hojas de cálculo, gráficas por computadora, multimedios, entretenimiento,

administración de base de datos y aplicaciones para finanzas personales o de negocios).

Aplicaciones web: llamadas “webapps”, esta categoría de software centrado en redes agrupa una amplia gama de aplicaciones. En su forma más sencilla, las webapps son poco más que un conjunto de archivos de hipertexto vinculados que presentan información con uso de texto y gráficas limitadas. Sin embargo, desde que surgió Web 2.0, las webapps están evolucionando hacia ambientes de cómputo sofisticados que no sólo proveen características aisladas, funciones de cómputo y contenido para el usuario final, sino que también están integradas con bases de datos corporativas y aplicaciones de negocios.

Software de inteligencia artificial: hace uso de algoritmos no numéricos para resolver problemas complejos que no son fáciles de tratar computacionalmente o con el análisis directo. Las aplicaciones en esta área incluyen robótica, sistemas expertos, reconocimiento de patrones (imagen y voz), redes neurales artificiales, demostración de teoremas y juegos.

c. Software heredado

Los sistemas de software heredado fueron desarrollados hace varias décadas y han sido modificados de manera continua para que satisfagan los cambios en los requerimientos de los negocios y plataformas de computación. La proliferación de tales sistemas es causa de dolores de cabeza para las organizaciones grandes, a las que resulta costoso mantenerlos y riesgoso hacerlos evolucionar.

Desafortunadamente, en ocasiones hay otra característica presente en el software heredado: mala calidad. Hay veces en las que los sistemas heredados tienen diseños que no son susceptibles de extenderse, código confuso, documentación mala o inexistente, casos y resultados de pruebas que nunca se archivaron, una historia de los cambios mal administrada, la lista es muy larga. A pesar de esto, dichos sistemas dan apoyo a las “funciones básicas del negocio y son indispensables para éste”. ¿Qué hacer?

La única respuesta razonable es: hacer nada, al menos hasta que el sistema heredado tenga un cambio significativo. Si el software heredado satisface las necesidades de sus usuarios y corre de manera confiable, entonces no falla ni necesita repararse. Sin embargo, conforme pase el tiempo será frecuente que los sistemas de software evolucionen por una o varias de las siguientes razones:

- El software debe adaptarse para que cumpla las necesidades de los nuevos ambientes del cómputo y de la tecnología.
- El software debe ser mejorado para implementar nuevos requerimientos del negocio.
- El software debe ampliarse para que sea operable con otros sistemas o bases de datos modernos.
- La arquitectura del software debe rediseñarse para hacerla viable dentro de un ambiente de redes.

2.1.12. Ingeniería del software

Sommerville (2005) indica que el software se ha incrustado profundamente en casi todos los aspectos de nuestras vidas y, como consecuencia, el número de personas que tienen interés en las características y funciones que brinda una aplicación específica ha crecido en forma notable. Cuando ha de construirse una aplicación nueva o sistema incrustado, deben escucharse muchas opiniones. Y en ocasiones parece que cada una de ellas tiene una idea un poco distinta de cuáles características y funciones debiera tener el software. Se concluye que debe hacerse un esfuerzo concertado para entender el problema antes de desarrollar una aplicación de software.

- Los requerimientos de la tecnología de la información que demandan los individuos, negocios y gobiernos se hacen más complejos con cada año que pasa. En la actualidad, grandes equipos de personas crean programas de cómputo que antes eran elaborados por un solo individuo. El software sofisticado, que alguna vez se implementó en un ambiente de cómputo predecible y autocontenido, hoy en día se halla incrustado en el interior de todo, desde la electrónica de

consumo hasta dispositivos médicos o sistemas de armamento. La complejidad de estos nuevos sistemas y productos basados en computadora demanda atención cuidadosa a las interacciones de todos los elementos del sistema. Se concluye que el diseño se ha vuelto una actividad crucial.

- Los individuos, negocios y gobiernos dependen cada vez más del software para tomar decisiones estratégicas y tácticas, así como para sus operaciones y control cotidianos. Si el software falla, las personas y empresas grandes pueden experimentar desde un inconveniente menor hasta fallas catastróficas. Se concluye que el software debe tener alta calidad.
- A medida que aumenta el valor percibido de una aplicación específica se incrementa la probabilidad de que su base de usuarios y longevidad también crezcan. Conforme se extienda su base de usuarios y el tiempo de uso, las demandas para adaptarla y mejorarla también crecerán. Se concluye que el software debe tener facilidad para recibir mantenimiento.

Estas realidades simples llevan a una conclusión: debe hacerse ingeniería con el software en todas sus formas y a través de todos sus dominios de aplicación.

Aunque cientos de autores han desarrollado definiciones personales de la ingeniería de software, la propuesta por Fritz Bauer (1969) en la conferencia fundamental sobre el tema todavía sirve como base para el análisis:

La ingeniería de software es el establecimiento y uso de principios fundamentales de la ingeniería con objeto de desarrollar en forma económica software que sea confiable y que trabaje con eficiencia en máquinas reales.

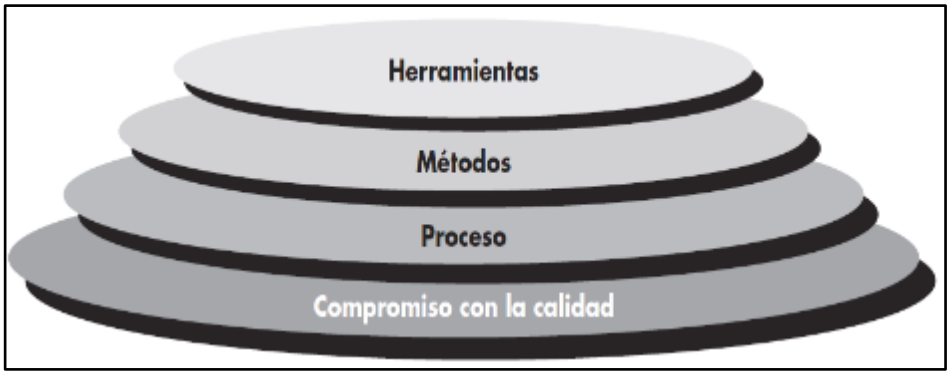

Figura 4. Capas de la ingeniería de software.

La ingeniería de software es una tecnología con varias capas. Como se aprecia en la figura anterior cualquier enfoque de ingeniería (incluso la de software) debe basarse en un compromiso organizacional con la calidad. La administración total de la calidad, Six Sigma y otras filosofías similares alimentan la cultura de mejora continua, y es esta cultura la que lleva en última instancia al desarrollo de enfoques cada vez más eficaces de la ingeniería de software. El fundamento en el que se apoya la ingeniería de software es el compromiso con la calidad.

2.1.13. El proceso del software

Pressman (2003) indica un proceso es un conjunto de actividades, acciones y tareas que se ejecutan cuando va a crearse algún producto del trabajo. Una actividad busca lograr un objetivo amplio (por ejemplo, comunicación con los participantes) y se desarrolla sin importar el dominio de la aplicación, tamaño del proyecto, complejidad del esfuerzo o grado de rigor con el que se usará la ingeniería de software. Una acción (diseño de la arquitectura) es un conjunto de tareas que producen un producto importante del trabajo (por ejemplo, un modelo del diseño de la arquitectura). Una tarea se centra en un objetivo pequeño pero bien definido (por ejemplo, realizar una prueba unitaria) que produce un resultado tangible.

En el contexto de la ingeniería de software, un proceso no es una prescripción rígida de cómo elaborar software de cómputo. Por el contrario, es un enfoque adaptable que permite que las personas que hacen el trabajo (el equipo de software) busquen y elijan el conjunto apropiado de acciones y tareas para el trabajo. Se busca siempre entregar el software en forma

oportuna y con calidad suficiente para satisfacer a quienes patrocinaron su creación y a aquellos que lo usarán.

La estructura del proceso establece el fundamento para el proceso completo de la ingeniería de software por medio de la identificación de un número pequeño de actividades estructurales que sean aplicables a todos los proyectos de software, sin importar su tamaño o complejidad. Además, la estructura del proceso incluye un conjunto de actividades sombrilla que son aplicables a través de todo el proceso del software. Una estructura de proceso general para la ingeniería de software consta de cinco actividades:

Comunicación: antes de que comience cualquier trabajo técnico, tiene importancia crítica comunicarse y colaborar con el cliente (y con otros participantes). Se busca entender los objetivos de los participantes respecto del proyecto, y reunir los requerimientos que ayuden a definir las características y funciones del software.

Planeación: cualquier viaje complicado se simplifica si existe un mapa. Un proyecto de software es un viaje difícil, y la actividad de planeación crea un "mapa" que guía al equipo mientras viaja. El mapa llamado plan del proyecto de software define el trabajo de ingeniería de software al describir las tareas técnicas por realizar, los riesgos probables, los recursos que se requieren, los productos del trabajo que se obtendrán y una programación de las actividades.

Modelado: ya sea usted diseñador de paisaje, constructor de puentes, ingeniero aeronáutico, carpintero o arquitecto, a diario trabaja con modelos. Crea un "bosquejo" del objeto por hacer a fin de entender el panorama general cómo se verá arquitectónicamente, cómo ajustan entre sí las partes constituyentes y muchas características más. Si se requiere, refina el bosquejo con más y más detalles en un esfuerzo por comprender mejor el problema y cómo resolverlo. Un ingeniero de software hace lo mismo al crear modelos a fin de entender mejor los requerimientos del software y el diseño que los satisfará.

Construcción: esta actividad combina la generación de código (ya sea manual o automatizada) y las pruebas que se requieren para descubrir errores en éste. Despliegue. El software (como entidad completa o como un incremento parcialmente terminado) se entrega al consumidor que lo evalúa y que le da retroalimentación, misma que se basa en dicha evaluación.

Figura 5. Capas de la ingeniería de software.

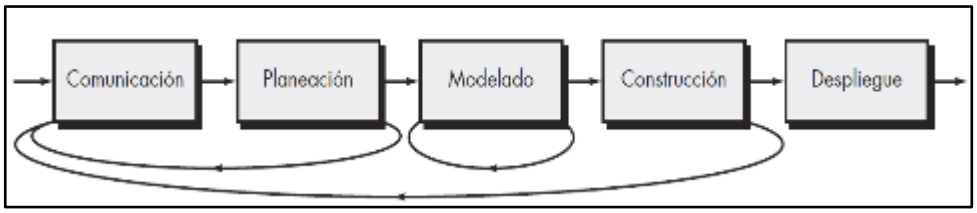

Figura 6. Flujo de proceso iterativo.

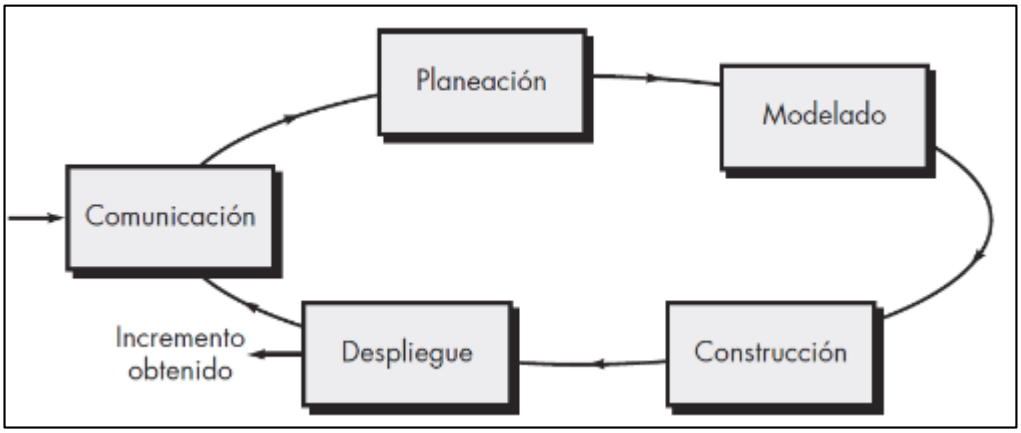

Figura 7. Flujo de proceso evolutivo.

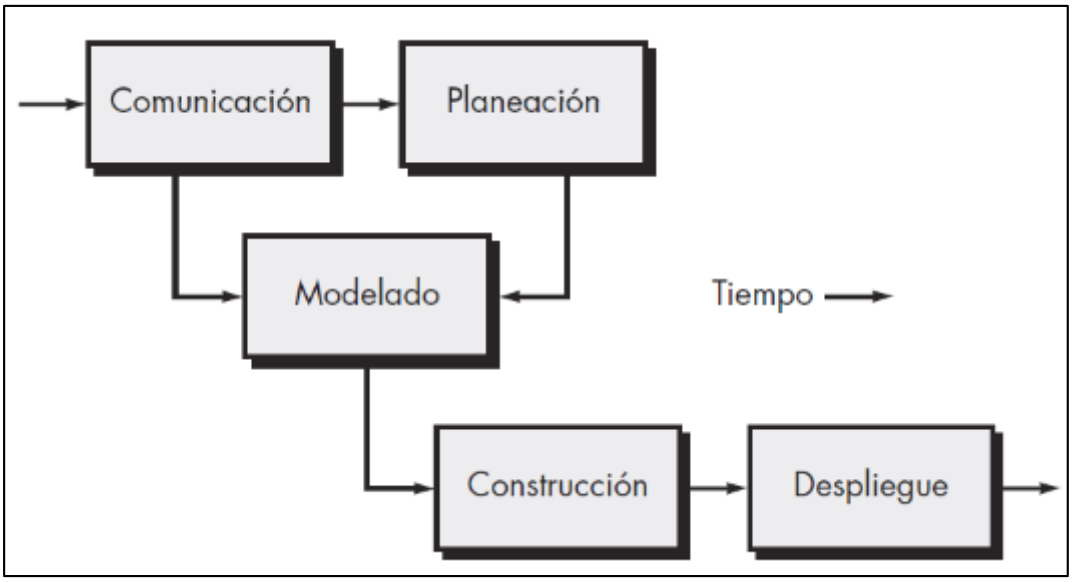

Figura 8. Flujo de proceso paralelo.

a. Evaluación y mejora del proceso

La existencia de un proceso del software no es garantía de que el software se entregue a tiempo, que satisfaga las necesidades de los consumidores o que tenga las características técnicas que conducirán a características de calidad de largo plazo. Los patrones de proceso deben acoplarse con una práctica sólida de ingeniería de software Además, el proceso en sí puede evaluarse para garantizar que cumple con ciertos criterios de proceso básicos que se haya demostrado que son esenciales para el éxito de la ingeniería de software.

En las últimas décadas se han propuesto numerosos enfoques para la evaluación y mejora de un proceso del software:

Método de evaluación del estándar CMMI para el proceso de mejora (**SCAMPI,** por sus siglas en inglés): proporciona un modelo de cinco fases para evaluar el proceso: inicio, diagnóstico, establecimiento, actuación y aprendizaje. El método SCAMPI emplea el SEI CMMI como la base de la evaluación.

Evaluación basada en CMM para la mejora del proceso interno (CBA IPI, por sus siglas en inglés): proporciona una técnica de diagnóstico para

evaluar la madurez relativa de una organización de software; usa el SEI CMM como la base de la evaluación.

SPICE (ISO/IEC 15504): estándar que define un conjunto de requerimientos para la evaluación del proceso del software. El objetivo del estándar es ayudar a las organizaciones a desarrollar una evaluación objetiva de cualquier proceso del software definido.

ISO9001:2000 para software: estándar genérico que se aplica a cualquier organización que desee mejorar la calidad general de los productos, sistemas o servicios que proporciona. Por tanto, el estándar es directamente aplicable a las organizaciones y compañías de software.

b. Modelos de proceso prescriptivo

Modelo de la cascada

Hay veces en las que los requerimientos para cierto problema se comprenden bien: cuando el trabajo desde la comunicación hasta el despliegue fluye en forma razonablemente lineal. Esta situación se encuentra en ocasiones cuando deben hacerse adaptaciones o mejoras bien definidas a un sistema ya existente (por ejemplo, una adaptación para software de contabilidad que es obligatorio hacer debido a cambios en las regulaciones gubernamentales). También ocurre en cierto número limitado de nuevos esfuerzos de desarrollo, pero sólo cuando los requerimientos están bien definidos y tienen una estabilidad razonable.

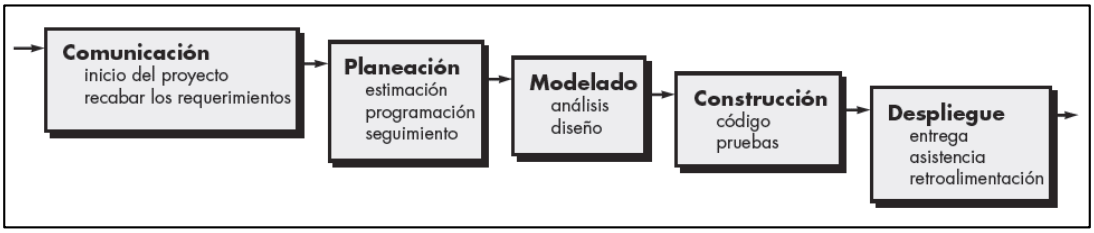

Figura 9. Modelo de la cascada.

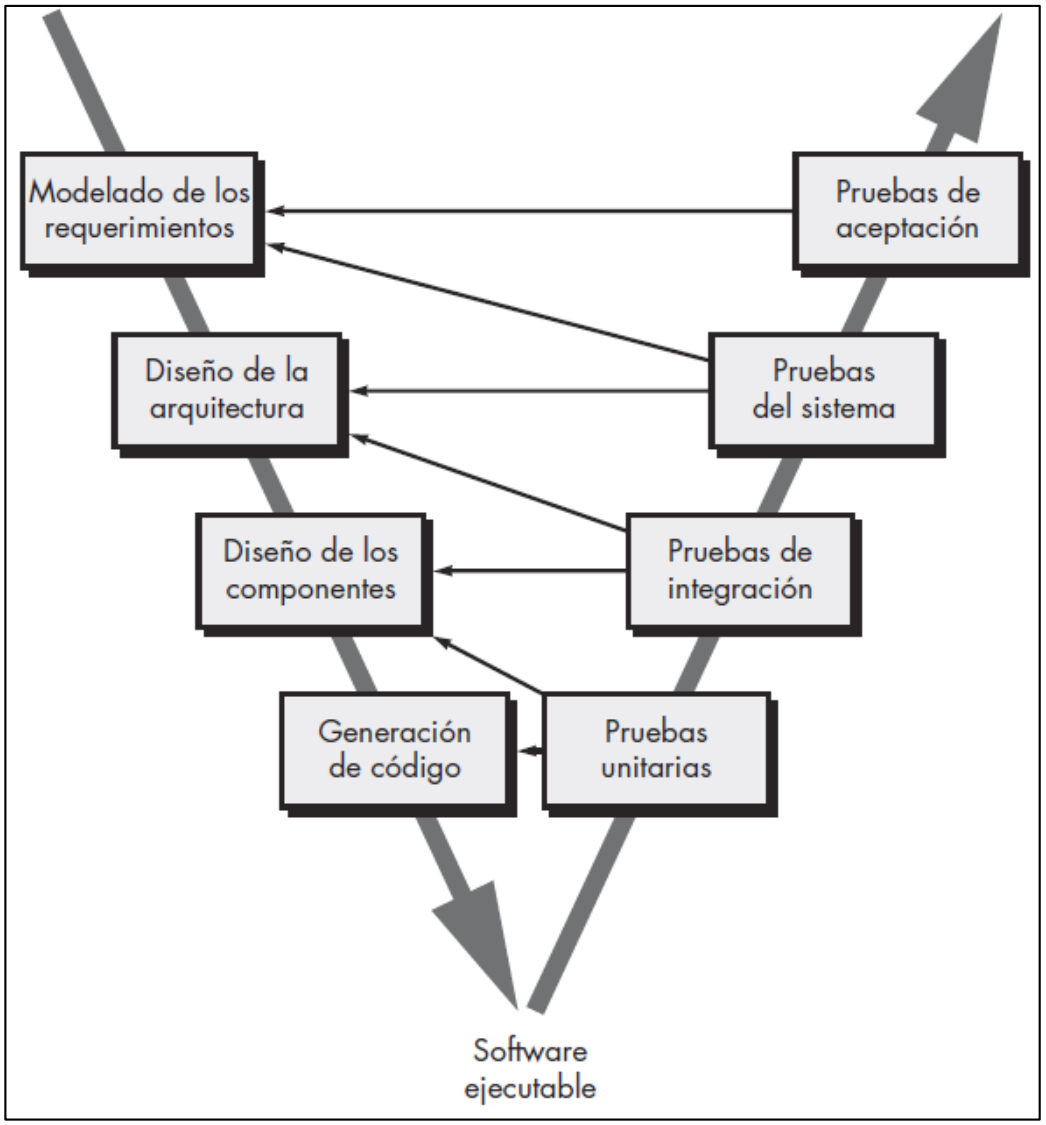

Figura 10. El modelo en V.

Modelos de proceso incremental

Hay muchas situaciones en las que los requerimientos iniciales del software están razonablemente bien definidos, pero el alcance general del esfuerzo de desarrollo imposibilita un proceso lineal. Además, tal vez haya una necesidad imperiosa de dar rápidamente cierta funcionalidad limitada de software a los usuarios y aumentarla en las entregas posteriores de software. En tales casos, se elige un modelo de proceso diseñado para producir el software en incrementos.

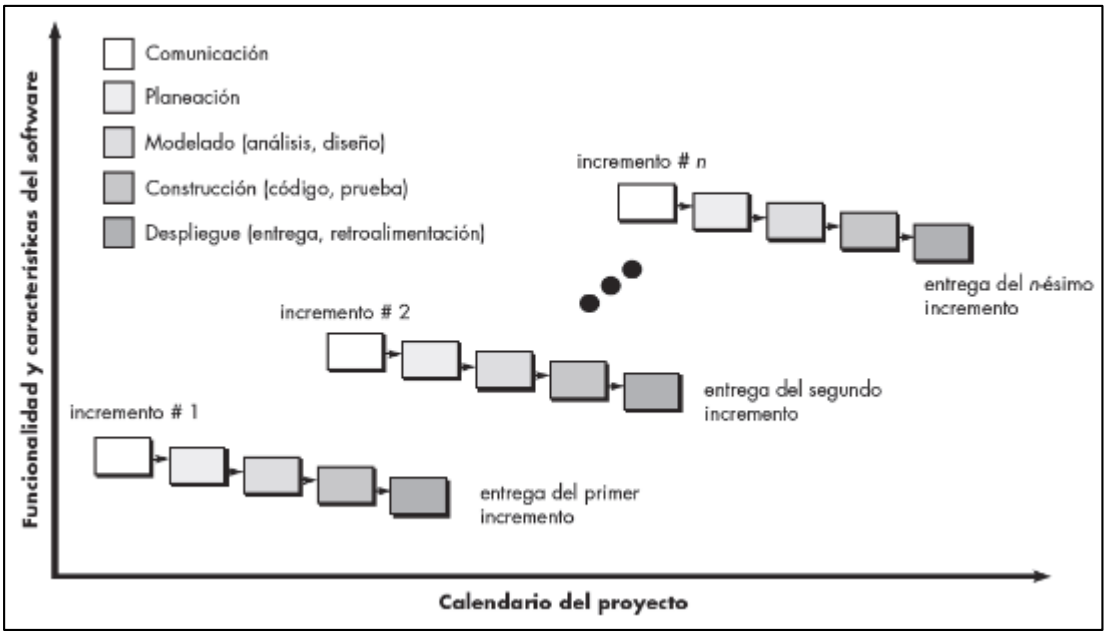

Figura 11. El modelo Incremental.

Modelos de proceso evolutivo

El software, como todos los sistemas complejos, evoluciona en el tiempo. Es frecuente que los requerimientos del negocio y del producto cambien conforme avanza el desarrollo, lo que hace que no sea realista trazar una trayectoria rectilínea hacia el producto final; los plazos apretados del mercado hacen que sea imposible la terminación de un software perfecto, pero debe lanzarse una versión limitada a fin de aliviar la presión de la competencia o del negocio; se comprende bien el conjunto de requerimientos o el producto básico, pero los detalles del producto o extensiones del sistema aún están por definirse. En estas situaciones y otras parecidas se necesita un modelo de proceso diseñado explícitamente para adaptarse a un producto que evoluciona con el tiempo.

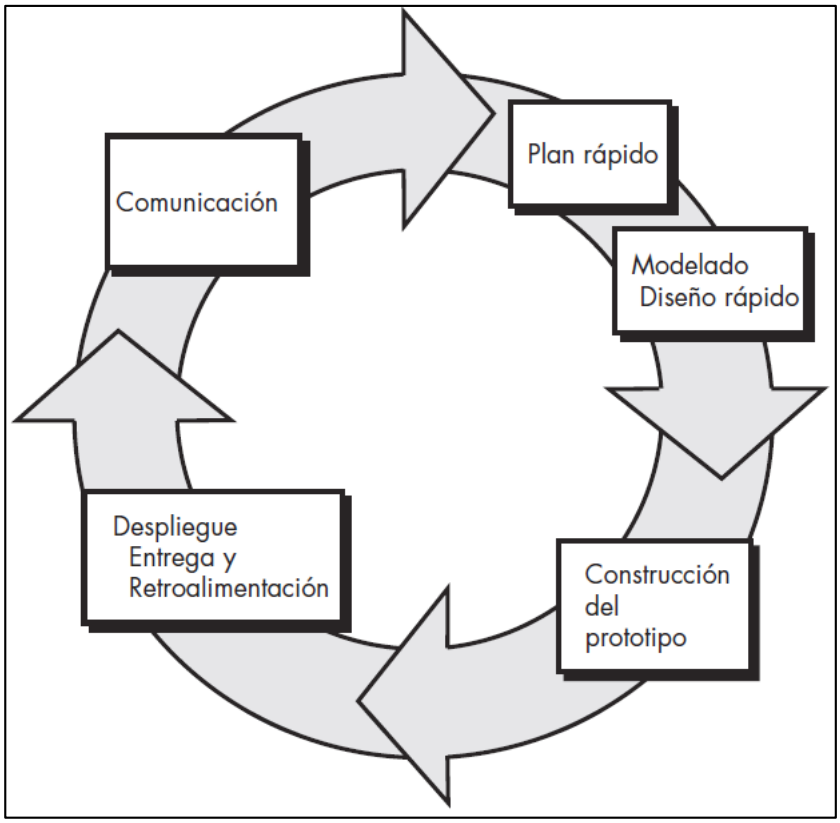

Figura 12. El paradigma de hacer prototipos.

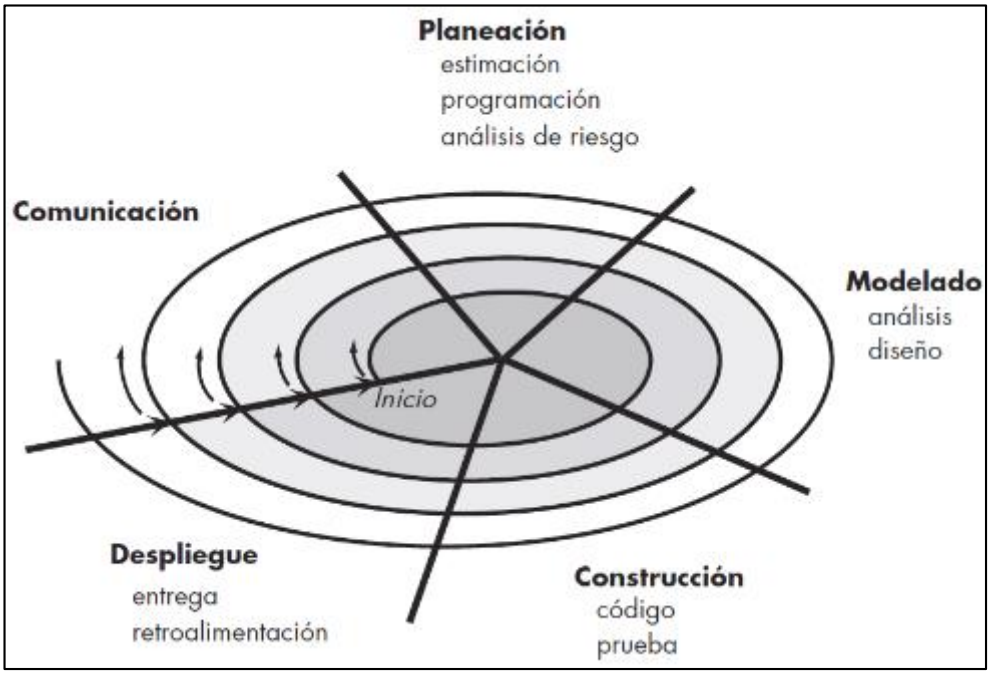

Figura 13. El paradigma de hacer prototipos.

Modelos concurrentes

El modelo de desarrollo concurrente, en ocasiones llamado ingeniería concurrente, permite que un equipo de software represente elementos iterativos y concurrentes de cualquiera de los modelos de proceso descritos en este capítulo. Por ejemplo, la actividad de modelado definida para el modelo espiral se logra por medio de invocar una o más de las siguientes acciones de software: hacer prototipos, análisis y diseño.

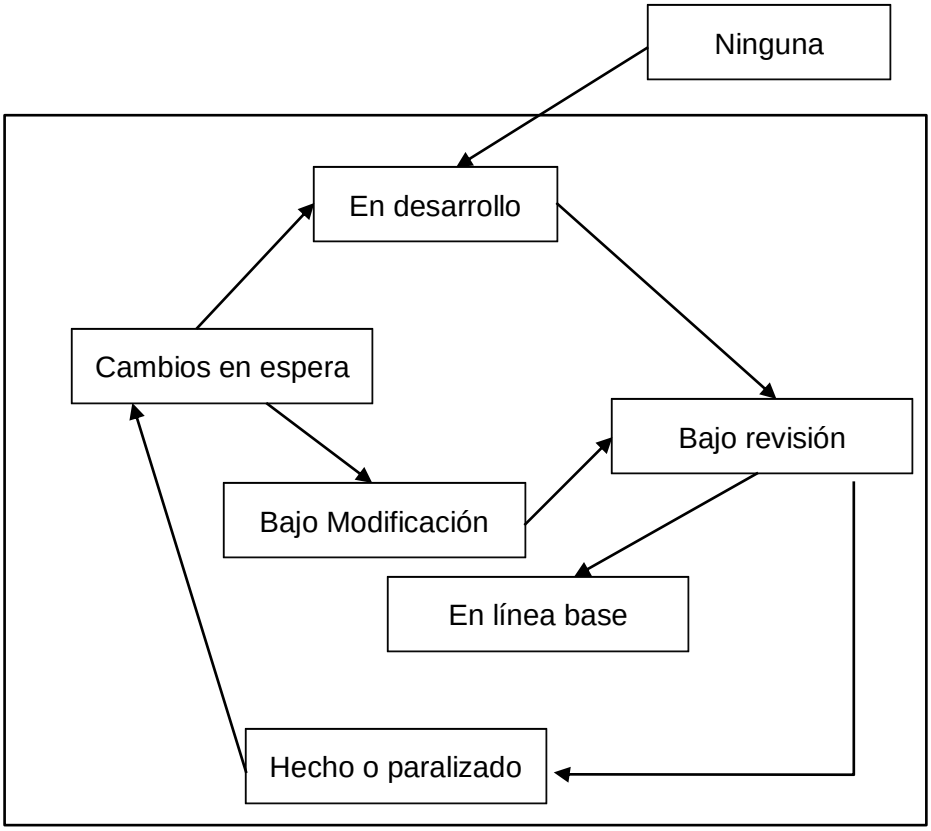

Figura 14. Un elemento del modelo de proceso concurrente.

c. El proceso unificado

En cierto modo, el proceso unificado es un intento por obtener los mejores rasgos y características de los modelos tradicionales del proceso del software, pero en forma que implemente muchos de los mejores principios del desarrollo ágil de software. El proceso unificado reconoce la importancia de la comunicación con el cliente y los métodos directos para describir su punto de vista respecto de un sistema (el caso de uso). Hace énfasis en la importancia

de la arquitectura del software y “ayuda a que el arquitecto se centre en las metas correctas, tales como que sea comprensible, permita cambios futuros y la reutilización”: Sugiere un flujo del proceso iterativo e incremental, lo que da la sensación evolutiva que resulta esencial en el desarrollo moderno del software.

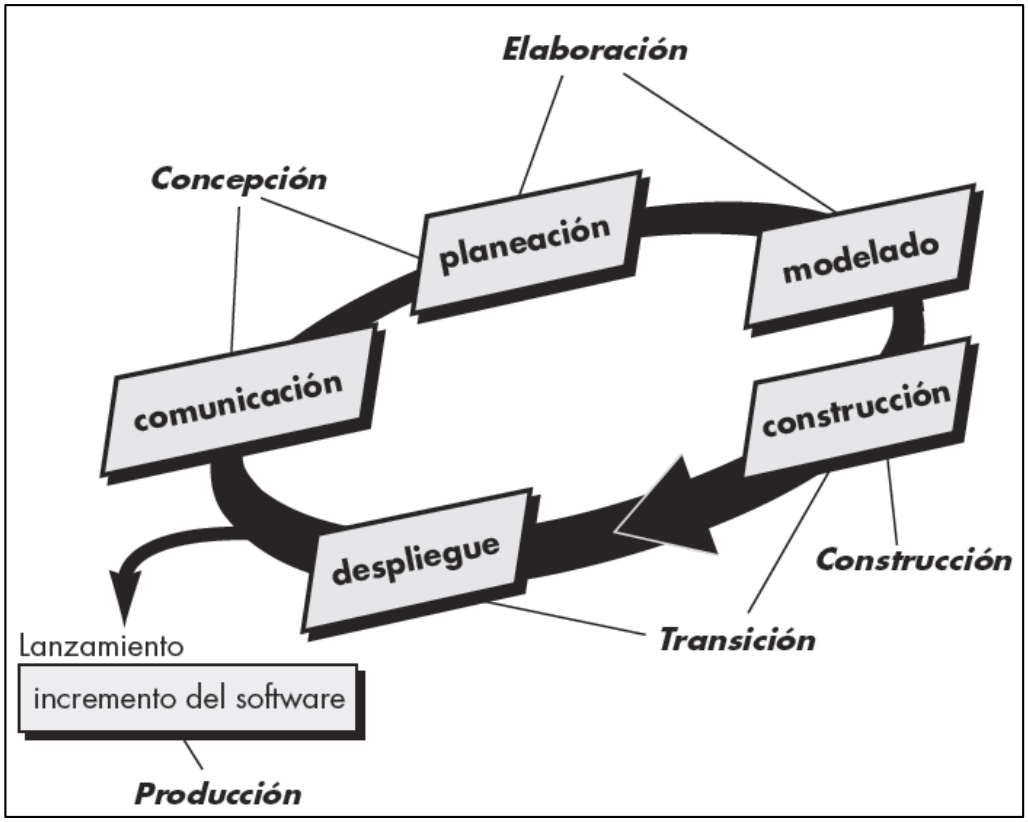

Figura 15. El proceso unificado.

Seguimiento y control del proyecto de software

Permite que el equipo de software evalúe el progreso comparándolo con el plan del proyecto y tome cualquier acción necesaria para apegarse a la programación de actividades.

Aseguramiento de la calidad del software

Define y ejecuta las actividades requeridas para garantizar la calidad del software.

d. Comprensión de los requerimientos

Entender los requerimientos de un problema es una de las tareas más difíciles que enfrenta el ingeniero de software. Cuando se piensa por primera vez, no parece tan difícil desarrollar un entendimiento claro de los requerimientos. Después de todo, ¿acaso no sabe el cliente lo que se necesita? ¿No deberían tener los usuarios finales una buena comprensión de las características y funciones que le darán un beneficio? Sorprendentemente, en muchas instancias la respuesta a estas preguntas es “no”. E incluso si los clientes y los usuarios finales explican sus necesidades, éstas cambiarán mientras se desarrolla el proyecto.

Es la peor de las pesadillas. Un cliente entra a la oficina, toma asiento, lo mira a uno fijamente a los ojos y dice: “Sé que cree que entiende lo que digo, pero lo que usted no entiende es que lo que digo no es lo que quiero decir.” Invariablemente, esto pasa cuando ya está avanzado el proyecto, después de que se han hecho compromisos con los plazos de entrega, que hay reputaciones en juego y mucho dinero invertido.

Ingeniería de requerimientos

El diseño y construcción de software de computadora es difícil, creativo y sencillamente divertido. En realidad, elaborar software es tan atractivo que muchos desarrolladores de software quieren ir directo a él antes de haber tenido el entendimiento claro de lo que se necesita. Argumentan que las cosas se aclararán a medida que lo elaboren, que los participantes en el proyecto podrán comprender sus necesidades sólo después de estudiar las primeras iteraciones del software, que las cosas cambian tan rápido que cualquier intento de entender los requerimientos en detalle es una pérdida de tiempo, que las utilidades salen de la producción de un programa que funcione y que todo lo demás es secundario. Lo que hace que estos argumentos sean tan seductores es que tienen algunos elementos de verdad. Pero todos son erróneos y pueden llevar un proyecto de software al fracaso.

Concepción: ¿Cómo inicia un proyecto de software? ¿Existe un solo evento que se convierte en el catalizador de un nuevo sistema o producto

basado en computadora o la necesidad evoluciona en el tiempo? No hay respuestas definitivas a estas preguntas. En ciertos casos, una conversación casual es todo lo que se necesita para desencadenar un trabajo grande de ingeniería de software.

Indagación: en verdad que parece muy simple: preguntar al cliente, a los usuarios y a otras personas cuáles son los objetivos para el sistema o producto, qué es lo que va a lograrse, cómo se ajusta el sistema o producto a las necesidades del negocio y, finalmente, cómo va a usarse el sistema o producto en las operaciones cotidianas. Pero no es simple: es muy difícil.

Elaboración: la información obtenida del cliente durante la concepción e indagación se expande y refina durante la elaboración. Esta tarea se centra en desarrollar un modelo refinado de los requerimientos que identifique distintos aspectos de la función del software, su comportamiento e información.

La elaboración está motivada por la creación y mejora de escenarios de usuario que describan cómo interactuará el usuario final (y otros actores) con el sistema. Cada escenario de usuario se enuncia con sintaxis apropiada para extraer clases de análisis, que son entidades del dominio del negocio visibles para el usuario final. Se definen los atributos de cada clase de análisis y se identifican los servicios4 que requiere cada una de ellas. Se identifican las relaciones y colaboración entre clases, y se producen varios diagramas adicionales.

Negociación: no es raro que los clientes y usuarios pidan más de lo que puede lograrse dado lo limitado de los recursos del negocio. También es relativamente común que distintos clientes o usuarios propongan requerimientos conflictivos con el argumento de que su versión es “esencial para nuestras necesidades especiales”.

Especificación: en el contexto de los sistemas basados en computadora (y software), el término especificación tiene diferentes significados para distintas personas. Una especificación puede ser un

documento escrito, un conjunto de modelos gráficos, un modelo matemático formal, un conjunto de escenarios de uso, un prototipo o cualquier combinación de éstos.

Validación: la calidad de los productos del trabajo que se generan como consecuencia de la ingeniería de los requerimientos se evalúa durante el paso de validación. La validación de los requerimientos analiza la especificación a fin de garantizar que todos ellos han sido enunciados sin ambigüedades; que se detectaron y corrigieron las inconsistencias, las omisiones y los errores, y que los productos del trabajo se presentan conforme a los estándares establecidos para el proceso, el proyecto y el producto.

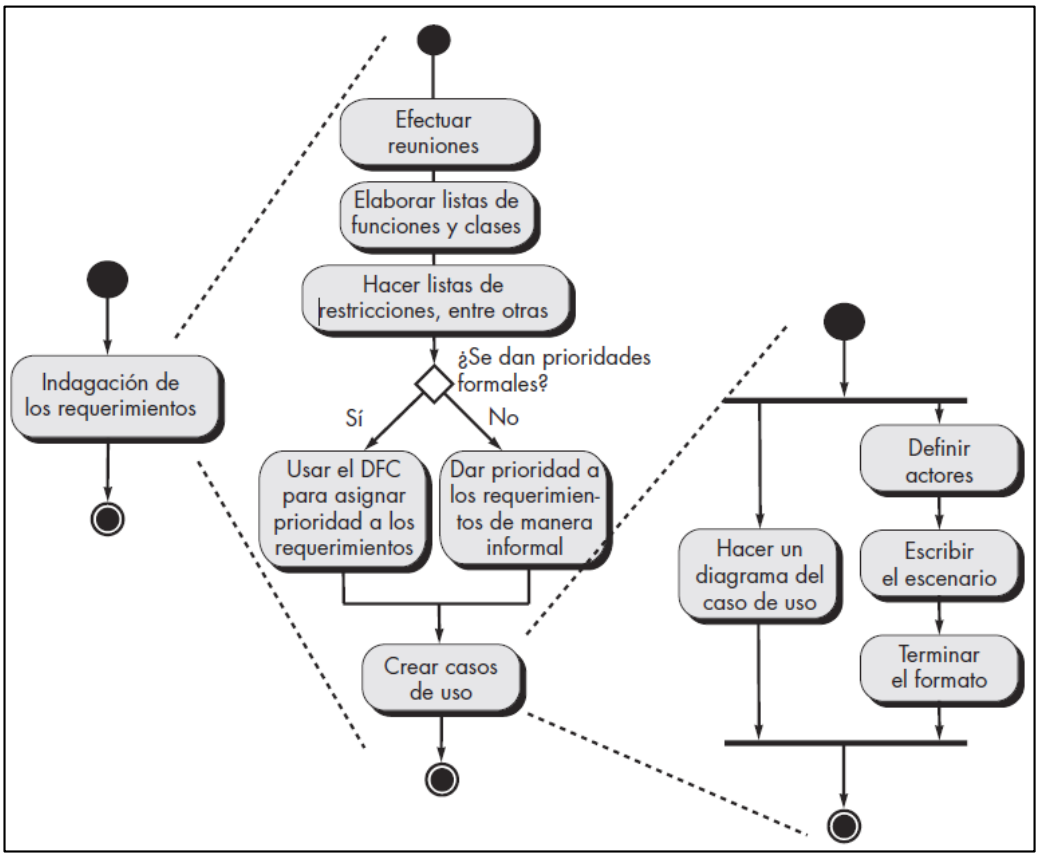

Figura 16. Diagramas de actividad del UML para indagar los requerimientos.

e. Diseño de la interfaz de usuario

Vivimos en un mundo de productos de alta tecnología, y virtualmente todos ellos electrónica para el consumidor, equipo industrial, sistemas

corporativos, sistemas militares, software de computadoras personales y webapps requieren interacción humana. Si un producto ha de alcanzar el éxito, debe tener buena usabilidad: medición cualitativa de la facilidad y eficiencia con la que un humano emplea las funciones y características que ofrece el producto de alta tecnología.

La usabilidad importa, ya sea que una interfaz haya sido diseñada para un reproductor de música digital o para el sistema de control de armas de un avión de combate. Si los mecanismos de la interfaz están bien diseñados, el usuario se desliza por la interacción a un ritmo suave que hace que el trabajo se realice sin esfuerzo. Pero si la interfaz fue mal concebida, el usuario avanza y retrocede, y el resultado final es frustración y poca eficiencia en el trabajo.

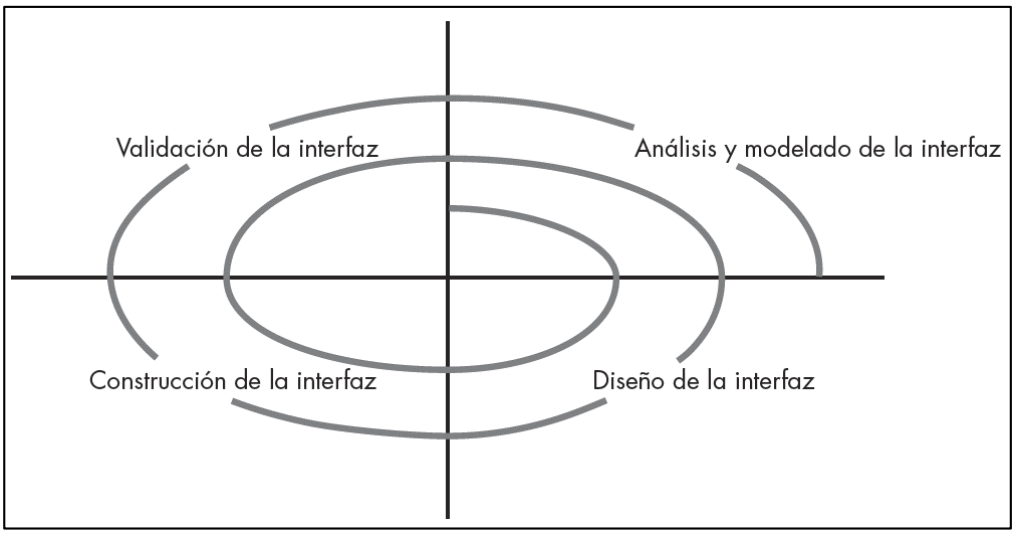

Figura 17. Diagramas de actividad del UML para indagar los requerimientos.

2.1.14. Diseño en el contexto de la ingeniería de software

Pressman (2003) menciona que el diseño de software se ubica en el área técnica de la ingeniería de software y se aplica sin importar el modelo del proceso que se utilice. El diseño del software comienza una vez que se han analizado y modelado los requerimientos, es la última acción de la ingeniería de software dentro de la actividad de modelado y prepara la etapa de construcción (generación y prueba de código).

Cada uno de los elementos del modelo de requerimientos proporciona información necesaria para crear los cuatro modelos de diseño necesarios

para la especificación completa del diseño. El trabajo de diseño es alimentado por el modelo de requerimientos, manifestado por elementos basados en el escenario, en la clase, orientados al flujo, y del comportamiento. El empleo de la notación y de los métodos de diseño estudiados en los últimos capítulos produce diseños de los datos o clases, de la arquitectura, de la interfaz y de los componentes.

El diseño de datos o clases transforma los modelos de clases en realizaciones de clases de diseño y en las estructuras de datos que se requieren para implementar el software. Los objetos y relaciones definidos en el diagrama CRC y el contenido detallado de los datos ilustrado por los atributos de clase y otros tipos de notación dan la base para el diseño de los datos. Parte del diseño de clase puede llevarse a cabo junto con el diseño de la arquitectura del software. Un diseño más detallado de las clases tiene lugar cuando se diseña cada componente del software.

El diseño de la arquitectura define la relación entre los elementos principales de la estructura del software, los estilos y patrones de diseño de la arquitectura que pueden usarse para alcanzar los requerimientos definidos por el sistema y las restricciones que afectan la forma en la que se implementa la arquitectura. La representación del diseño de la arquitectura - el marco de un sistema basado en computadora se obtiene del modelo de los requerimientos.

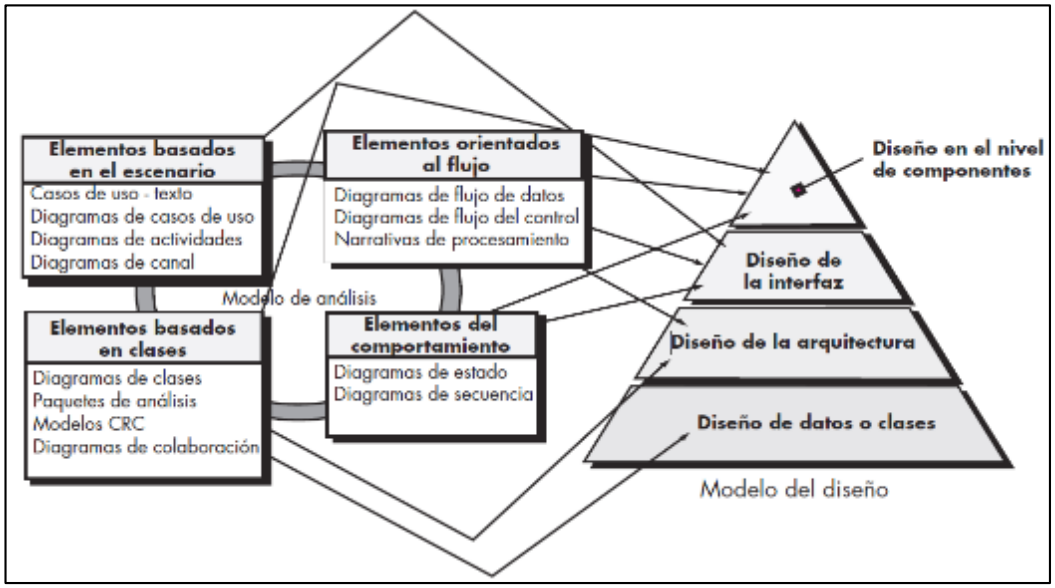

Figura 18. Diagrama de tareas para captura y análisis de requisitos.

El diseño de la interfaz describe la forma en la que el software se comunica con los sistemas que interactúan con él y con los humanos que lo utilizan. Una interfaz implica un flujo de información (por ejemplo, datos o control) y un tipo específico de comportamiento. Entonces, los modelos de escenarios de uso y de comportamiento dan mucha de la información requerida para diseñar la interfaz.

El diseño en el nivel de componente transforma los elementos estructurales de la arquitectura del software en una descripción de sus componentes en cuanto a procedimiento. La información obtenida a partir de los modelos basados en clase, flujo y comportamiento sirve como la base para diseñar los componentes.

2.1.15. Arquitectura del software

Pressman (2003) indica que la arquitectura del software alude a “la estructura general de éste y a las formas en las que ésta da integridad conceptual a un sistema”. En su forma más sencilla, la arquitectura es la estructura de organización de los componentes de un programa (módulos), la forma en la que éstos interactúan y la estructura de datos que utilizan. Sin embargo, en un sentido más amplio, los componentes se generalizan para que representen los elementos de un sistema grande y sus interacciones. Una meta del diseño del software es obtener una aproximación arquitectónica de un sistema. Ésta sirve como estructura a partir de la cual se realizan las actividades de diseño más detalladas. Un conjunto de patrones arquitectónicos permite que el ingeniero de software resuelva problemas de diseño comunes.

Pressman (2003) describen un conjunto de propiedades que deben especificarse como parte del diseño de la arquitectura:

Propiedades estructurales: este aspecto de la representación del diseño arquitectónico define los componentes de un sistema (módulos,

objetos, filtros, etc.) y la manera en la que están agrupados e interactúan unos con otros. Por ejemplo, los objetos se agrupan para que encapsulen tanto datos como el procedimiento que los manipula e interactúen invocando métodos.

Propiedades extrafuncionales: la descripción del diseño arquitectónico debe abordar la forma en la que la arquitectura del diseño satisface los requerimientos de desempeño, capacidad, confiabilidad, seguridad y adaptabilidad, así como otras características del sistema.

Familias de sistemas relacionados: el diseño arquitectónico debe basarse en patrones repetibles que es común encontrar en el diseño de familias de sistemas similares. En esencia, el diseño debe tener la capacidad de reutilizar bloques de construcción arquitectónica.

Dada la especificación de estas propiedades, el diseño arquitectónico se representa con el uso de uno o más de varios modelos diferentes. Los modelos estructurales representan la arquitectura como un conjunto organizado de componentes del programa. Los modelos de marco aumentan el nivel de abstracción del diseño, al tratar de identificar patrones de diseño arquitectónico repetibles que se encuentran en tipos similares de aplicaciones. Los modelos dinámicos abordan los aspectos estructurales de la arquitectura del programa e indican cómo cambia la estructura o la configuración del sistema en función de eventos externos. Los modelos del proceso se centran en el diseño del negocio o proceso técnico al que debe dar acomodo el sistema. Por último, los modelos funcionales se usan para representar la jerarquía funcional de un sistema. Para representar estos modelos, se ha desarrollado cierto número de lenguajes de descripción arquitectónica. Aunque han sido propuestos muchos LDA diferentes, la mayoría tiene mecanismos para describir los componentes del sistema y la manera en la que se conectan entre sí. Debe observarse que hay un debate acerca del papel que tiene la arquitectura en el diseño. Algunos investigadores afirman que la obtención de la arquitectura del software debe separarse del diseño y

que ocurre entre las acciones de la ingeniería de requerimientos y las del diseño más convencional. Otros piensan que la definición de la arquitectura es parte integral del proceso de diseño.

¿Por qué es importante la arquitectura?

- Las representaciones de la arquitectura del software permiten la comunicación entre todas las partes (participantes) interesadas en el desarrollo de un sistema basado en computadora.
- La arquitectura resalta las primeras decisiones que tendrán un efecto profundo en todo el trabajo de ingeniería de software siguiente y, también importante, en el éxito último del sistema como entidad operacional.
- La arquitectura “constituye un modelo relativamente pequeño y asequible por la vía intelectual sobre cómo está estructurado el sistema y la forma en la que sus componentes trabajan juntos”.

El modelo del diseño de la arquitectura y los patrones arquitectónicos contenidos dentro de éste son transferibles. Es decir, los géneros, estilos y patrones arquitectónicos pueden aplicarse al diseño de otros sistemas y representan un conjunto de abstracciones que permite a los ingenieros de software describir la arquitectura en formas predecibles.

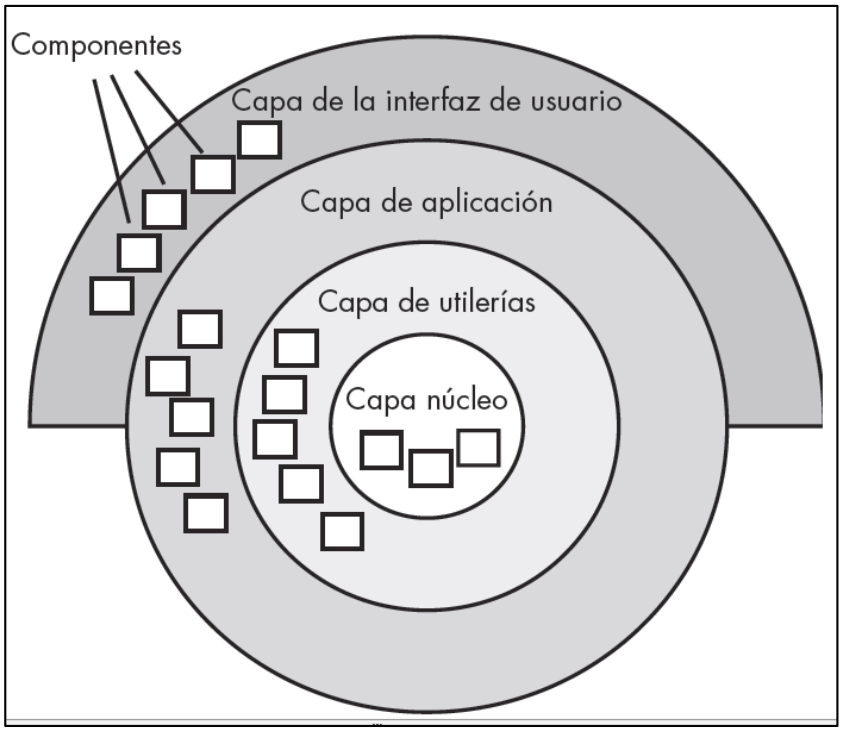

Figura 19. Arquitectura del software en capas.

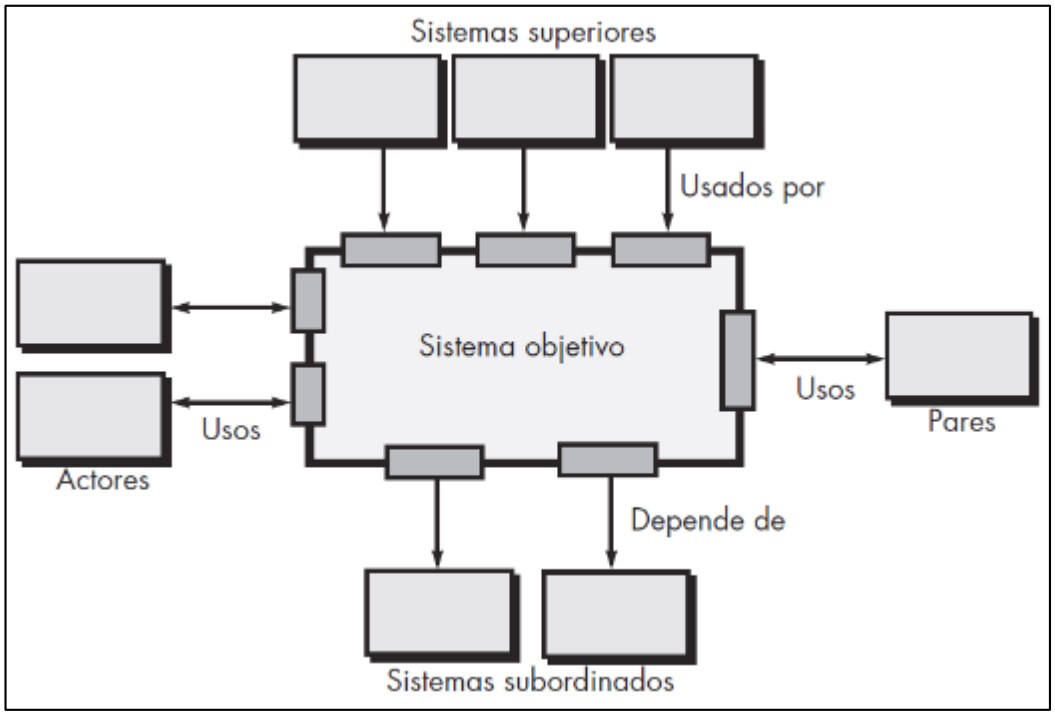

Figura 20. Diagrama de contexto arquitectónico.

2.1.16. Aseguramiento de la calidad del software

Calidad del software

Pressman (2003) menciona incluso los desarrolladores de software más experimentados estarán de acuerdo en que obtener software de alta calidad es una meta importante. Pero, ¿cómo se define la calidad del software? En el sentido más general se define como: Proceso eficaz de software que se aplica de manera que crea un producto útil que proporciona valor medible a quienes lo producen y a quienes lo utilizan. Hay pocas dudas acerca de que la definición anterior podría modificarse o ampliarse en un debate sin fin. Para propósitos de este autor en su libro, la misma sirve a fin de enfatizar tres puntos importantes:

1. Un proceso eficaz de software establece la infraestructura que da apoyo a cualquier esfuerzo de elaboración de un producto de software de alta calidad.
2. Un producto útil entrega contenido, funciones y características que el usuario final desea; sin embargo, de igual importancia es que entrega estos activos en forma confiable y libre de errores.
3. Al agregar valor para el productor y para el usuario de un producto, el software de alta calidad proporciona beneficios a la organización que lo produce y a la comunidad de usuarios finales.

Elementos de aseguramiento de la calidad del software

El aseguramiento de la calidad del software incluye un rango amplio de preocupaciones y actividades que se centran en la administración de la calidad del software. Éstas se resumen como sigue:

Estándares: el IEEE, ISO y otras organizaciones que establecen estándares han producido una amplia variedad de ellos para ingeniería de software y documentos relacionados. Los estándares los adopta de manera voluntaria una organización de software o los impone el cliente u otros

participantes. El trabajo del ACS es asegurar que los estándares que se hayan adoptado se sigan, y que todos los productos del trabajo se apeguen a ellos.

Revisiones y auditorías: las revisiones técnicas son una actividad del control de calidad que realizan ingenieros de software para otros ingenieros de software. Su objetivo es detectar errores. Las auditorías son un tipo de revisión efectuada por personal de ACS con objeto de garantizar que se sigan los lineamientos de calidad en el trabajo de la ingeniería de software. Por ejemplo, una auditoría del proceso de revisión se efectúa para asegurar que las revisiones se lleven a cabo de manera que tengan la máxima probabilidad de descubrir errores.

Pruebas: las pruebas del software son una función del control de calidad que tiene un objetivo principal: detectar errores. El trabajo del ACS es garantizar que las pruebas se planeen en forma apropiada y que se realicen con eficiencia, de modo que la probabilidad de que logren su objetivo principal sea máxima.

Colección y análisis de los errores: la única manera de mejorar es medir cómo se está haciendo algo. El ACS reúne y analiza errores y datos acerca de los defectos para entender mejor cómo se cometen los errores y qué actividades de la ingeniería de software son más apropiadas para eliminarlos.

Administración del cambio: el cambio es uno de los aspectos que más irrumpe en cualquier proyecto de software. Si no se administra en forma adecuada, lleva a la confusión y ésta casi siempre genera mala calidad. El ACS asegura que se hayan instituido prácticas adecuadas de administración del cambio.

Educación: toda organización de software quiere mejorar sus prácticas de ingeniería de software. Un contribuyente clave de la mejora es la educación de los ingenieros de software, de sus gerentes y de otros participantes. La

organización de ACS lleva el liderazgo en la mejora del proceso de software y es clave para proponer y patrocinar programas educativos.

Administración de los proveedores: son tres las categorías de software que se adquieren a proveedores externos: paquetes contenidos en una caja (por ejemplo, Office, de Microsoft); un shell personalizado, que da una estructura básica, tipo esqueleto, que se adapta de manera única a las necesidades del comprador; y software contratado, que se diseña y construye especialmente a partir de especificaciones provistas por la organización cliente

Administración de la seguridad: con el aumento de los delitos cibernéticos y de las nuevas regulaciones gubernamentales respecto de la privacidad, toda organización de software debe instituir políticas para proteger los datos en todos los niveles, establecer cortafuegos de protección para las webapps y asegurar que el software no va a ser vulnerado in ternamente. El ACS garantiza que para lograr la seguridad del software, se utilicen el proceso y la tecnología apropiados.

Seguridad: debido a que el software casi siempre es un componente crucial de los sistemas humanos (como aplicaciones automotrices o aeronáuticas), la consecuencia de defectos ocultos puede ser catastrófica. El ACS es responsable de evaluar el efecto de las fallas del software y de dar los pasos que se requieren para disminuir el riesgo.

Administración de riesgos: aunque el análisis y la mitigación de riesgos, es asunto de los ingenieros de software, la organización del ACS garantiza que las actividades de administración de riesgos se efectúen en forma apropiada y que se establezcan planes de contingencia relacionados con los riesgos.

Las normas de calidad ISO 9000

Un sistema de aseguramiento de la calidad se define como la estructura organizacional, responsabilidades, procedimientos, procesos y recursos necesarios para implementar la administración de la calidad. Los sistemas de aseguramiento de la calidad se crean para ayudar a las organizaciones a asegurar que sus productos y servicios satisfagan las expectativas del consumidor gracias a que cumplan con sus especificaciones. Estos sistemas cubren una amplia variedad de actividades, que contemplan todo el ciclo de vida del producto, incluidos planeación, control, medición, pruebas e informes, así como la mejora de los niveles de calidad en todo el proceso de desarrollo y manufactura. La norma ISO 9000 describe en términos generales los elementos de aseguramiento de la calidad que se aplican a cualquier negocio, sin importar los productos o servicios ofrecidos.

Para registrarse en alguno de los modelos del sistema de aseguramiento de la calidad contenidos en la ISO 9000, por medio de auditores externos se revisan en detalle el sistema y las operaciones de calidad de una compañía, respecto del cumplimiento del estándar y de la operación eficaz. Después de un registro exitoso, el grupo de registro representado por los auditores emite un certificado para la compañía. Auditorías semestrales de supervisión aseguran el cumplimiento continuo de la norma.

2.1.17. Programación orientado a objetos

Rumbaugh (1999) indica que la programación orientada a objetos (POO, u OOP según sus siglas en inglés) es un paradigma de programación que viene a innovar la forma de obtener resultados. Los objetos manipulan los datos de entrada para la obtención de datos de salida específicos, donde cada objeto ofrece una funcionalidad especial. Muchos de los objetos pre-diseñados de los lenguajes de programación actuales permiten la agrupación en bibliotecas o librerías, sin embargo, muchos de estos lenguajes permiten al usuario la creación de sus propias bibliotecas. Está basada en varias técnicas, incluyendo herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento. Su uso se popularizó a principios de la

década de 1990. En la actualidad, existe una gran variedad de lenguajes de programación que soportan la orientación a objetos.

Origen

Los conceptos de la POO tienen origen en Simula 67, un lenguaje diseñado para hacer simulaciones, creado por Ole-Johan Dahl y Kristen Nygaard, del Centro de Cómputo Noruego en Oslo. En este centro se trabajaba en simulaciones de naves, que fueron confundidas por la explosión combinatoria de cómo las diversas cualidades de diferentes naves podían afectar unas a las otras. La idea surgió al agrupar los diversos tipos de naves en diversas clases de objetos, siendo responsable cada clase de objetos de definir sus "propios" datos y comportamientos. Fueron refinados más tarde en Smalltalk, desarrollado en Simula en Xerox PARC (cuya primera versión fue escrita sobre Basic) pero diseñado para ser un sistema completamente dinámico en el cual los objetos se podrían crear y modificar "sobre la marcha" (en tiempo de ejecución) en lugar de tener un sistema basado en programas estáticos.

La POO se fue convirtiendo en el estilo de programación dominante a mediados de los años 1980, en gran parte debido a la influencia de C++, una extensión del lenguaje de programación C. Su dominación fue consolidada gracias al auge de las interfaces gráficas de usuario, para las cuales la POO está particularmente bien adaptada. En este caso, se habla también de programación dirigida por eventos.

Características de la POO

Existe un acuerdo acerca de qué características contempla la "orientación a objetos". Las características siguientes son las más importantes:

a. **Abstracción:** denota las características esenciales de un objeto, donde se capturan sus comportamientos. Cada objeto en el sistema

sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar "cómo" se implementan estas características. Los procesos, las funciones o los métodos pueden también ser abstraídos, y, cuando lo están, una variedad de técnicas son requeridas para ampliar una abstracción. El proceso de abstracción permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real. La abstracción es clave en el proceso de análisis y diseño orientado a objetos, ya que mediante ella podemos llegar a armar un conjunto de clases que permitan modelar la realidad o el problema que se quiere atacar

b. **Encapsulamiento:** significa reunir todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión (diseño estructurado) de los componentes del sistema. Algunos autores confunden este concepto con el principio de ocultación, principalmente porque se suelen emplear conjuntamente.

c. **Polimorfismo:** comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre; al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. O, dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado. Cuando esto ocurre en "tiempo de ejecución", esta última característica se llama asignación tardía o asignación dinámica. Algunos lenguajes proporcionan medios más estáticos (en "tiempo de compilación") de polimorfismo, tales como las plantillas y la sobrecarga de operadores de C++.

d. **Herencia:** las clases no se encuentran aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento, permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo. Esto suele hacerse habitualmente agrupando los objetos en clases y estas en árboles o enrejados que reflejan un comportamiento común. Cuando un objeto hereda de más de una clase se dice que hay herencia múltiple; siendo de alta complejidad técnica por lo cual suele recurrirse a la herencia virtual para evitar la duplicación de datos.
e. **Modularidad:** se denomina "modularidad" a la propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes. Estos módulos se pueden compilar por separado, pero tienen conexiones con otros módulos. Al igual que la encapsulación, los lenguajes soportan la modularidad de diversas formas.
f. **Principio de ocultación:** cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una "interfaz" a otros objetos que especifica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas; solamente los propios métodos internos del objeto pueden acceder a su estado. Esto asegura que otros objetos no puedan cambiar el estado interno de un objeto de manera inesperada, eliminando efectos secundarios e interacciones inesperadas. Algunos lenguajes relajan esto, permitiendo un acceso directo a los datos internos del objeto de una manera controlada y limitando el grado de abstracción. La aplicación entera se reduce a un agregado o rompecabezas de objetos.

g. **Recolección de desechos:** la recolección de basura (garbage collection) es la técnica por la cual el entorno de objetos se encarga de destruir automáticamente, y por tanto desvincular la memoria asociada, los objetos que hayan quedado sin ninguna referencia a ellos. Esto significa que el programador no debe preocuparse por la asignación o liberación de memoria, ya que el entorno la asignará al crear un nuevo objeto y la liberará cuando nadie lo esté usando. En la mayoría de los lenguajes híbridos que se extendieron para soportar el Paradigma de Programación Orientada a Objetos como C++ u Object Pascal, esta característica no existe y la memoria debe desasignarse expresamente.

Algunos lenguajes orientados a objetos

Entre los lenguajes orientados a objetos se destacan los siguientes: **Visual Basic .NET,** ABAP, ABL, ActionScript, ActionScript, Ada, C++, C, C Sharp (C#), Clarion, Clipper, D, Object Pascal (Embarcadero Delphi), Gambas, GObject, Genie, Harbour, Eiffel, Fortran 90/95, Java, JavaScript, Lexico, Objective-C, Ocaml, Oz, R, Pauscal (en español), Perl, PHP, PowerBuilder, Python, Ruby, Self, Smalltalk10, Magik SmallWorld), Vala, VB.NET, Visual FoxPro, Visual Basic 6.0, Visual, DataFlex, Visual Objects, XBase++, DRP, Scala (Pressman, 2003).

Visual Basic .NET

Visual Basic .NET (VB.NET) es un lenguaje de programación orientado a objetos que se puede considerar una evolución de Visual Basic implementada sobre el framework .NET. Su introducción resultó muy controvertida, ya que debido a cambios significativos en el lenguaje VB.NET no es retro compatible con Visual Basic, pero el manejo de las instrucciones es similar a versiones anteriores de Visual Basic, facilitando así el desarrollo de aplicaciones más avanzadas con herramientas modernas. Para mantener eficacia en el desarrollo de las aplicaciones. La gran mayoría de programadores de VB.NET utilizan el entorno de desarrollo integrado Microsoft Visual Studio en alguna de sus versiones (desde el primer Visual

Studio .NET hasta Visual Studio .NET 2015, que es la última versión de Visual Studio para la plataforma .NET), aunque existen otras alternativas, como SharpDevelop (que además es libre). Al igual que con todos los lenguajes de programación basados en .NET, los programas escritos en VB .NET requieren el Framework .NET o Mono para ejecutarse (Pressman, 2003).

2.1.18. Metodología RUP

Mendoza (2008) menciona que la metodología RUP, llamada así por sus siglas en inglés Rational Unified Process, es un proceso para llevar a cabo el desarrollo de un software, la cual define quien, como, cuando y que debe hacerse en el proyecto. Su propósito es asegurar la producción de software de alta calidad que satisfaga las necesidades de los usuarios finales, ajustándose a un presupuesto y tiempo establecidos.

Es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Sin embargo, el Proceso Unificado es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos.

El proceso unificado de desarrollo es el resultado de más de 30 años de experiencia, con la unificación de técnicas de desarrollo y del trabajo de muchos metodologístas. El antecedente más importante de esta metodología se ubica en 1967 con la Metodología Ericsson (Ericsson Approach) elaborada por Ivar Jacobson, que fue una aproximación de desarrollo basada en componentes, y que introdujo el concepto de Caso de Uso. Entre los años de 1987 a 1995 Jacobson fundó la compañía “Objectory AB” y lanza el proceso de desarrollo Objectory (abreviación de Object Factory).

Posteriormente en 1995 “Rational Software Corporation” adquiere “Objectory AB” y entre 1995 y 1997 se desarrolla Rational Objectory Process (ROP) fruto del encuentro y fusión de Objectory 3.8 y del Enfoque Rational (Rational Approach) adoptando UML como lenguaje de modelado. Desde ese entonces y a la cabeza de Ivar Jacobson, Grady Booch, y James Rumbaugh, Rational Software desarrolló e incorporó diversos elementos para expandir

ROP, destacándose especialmente el flujo de trabajo conocido como modelado del negocio. Es así como en junio del 1998 se lanza Rational Unified Process.

RUP proporciona un enfoque disciplinado para asignar tareas y responsabilidades dentro de una organización del desarrollo, e intenta integrar todos los aspectos que deben tenerse en cuenta durante todo el ciclo de vida del software, con el fin de que pueda ser aplicable tanto en pequeños como en grandes proyectos de software.

Esta metodología de desarrollo también ofrece una forma coordinada de trabajar; proporciona una guía para ordenar las actividades que deben realizar el equipo de proyecto, dirige las tareas de cada desarrollador por separado y del equipo como un todo, especifica los artefactos que deben desarrollarse y ofrece criterios para el control y la medición de los productos y actividades del proyecto.

Adicionalmente, el Proceso Unificado describe los diversos pasos involucrados en la captura de los requerimientos y en el establecimiento de una guía arquitectónica, para diseñar y probar el sistema de acuerdo a los requerimientos y a la arquitectura realizada del mismo. Además, provee patrones, y describe qué entregables producir y cómo desarrollarlos. RUP es soportado por herramientas que automatizan entre otras cosas, el modelado visual, la administración de cambios y las pruebas.

2.1.18.1. Características principales de RUP

Mendoza (2008) indica las características esenciales de RUP son las siguientes:

a) Proceso dirigido por casos de usos

Los casos de uso son una técnica de captura de requisitos potenciales de un nuevo sistema o la actualización de un software ya existente. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un

objetivo específico. En otras palabras, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.

En RUP los Casos de Uso no son sólo una herramienta para especificar los requisitos funcionales del sistema. También guían su diseño, implementación y prueba, es decir, guían el proceso de desarrollo del sistema. Por lo tanto dirigido por casos de uso significa que el proceso de desarrollo sigue un hilo conductor que permite establecer trazabilidad entre los artefactos que son generados en las diferentes actividades de dicho proceso.

b) Proceso centrado en la arquitectura

El papel de la arquitectura software es similar al papel que juega la arquitectura en la construcción de edificios. El edificio se mira desde diferentes puntos de vista: estructura, servicios, plomería, electricidad, etc. Esto permite al constructor ver una imagen completa antes de que comience la construcción. Similarmente, la arquitectura en un sistema software se describe mediante diferentes vistas del sistema en construcción.

La arquitectura de un sistema es la organización o estructura de sus partes más relevantes, que permite tener una visión común entre todos los involucrados (desarrolladores y usuarios) y una perspectiva clara del sistema completo, necesaria para controlar el desarrollo.

La metodología RUP además de utilizar los casos de uso para guiar el proceso, presta especial atención al establecimiento temprano de una buena arquitectura que no se vea fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento. RUP asume que no existe un modelo único que cubra todos los aspectos del sistema. Por tal motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema.

La arquitectura involucra los aspectos estáticos y dinámicos más significativos del sistema. Esta surge de las necesidades de la empresa, como las perciben los usuarios y los inversores, y se refleja en los casos de uso. Sin embargo, también está influenciada por muchos otros factores, tales como la

plataforma en la que tiene que funcionar el software (arquitectura hardware, sistema operativo, sistema de gestión de base de datos, protocolo para comunicaciones en red), la disponibilidad de componentes reutilizables, consideraciones de instalación, sistemas heredados y requisitos no funcionales (por ejemplo rendimiento y confiabilidad).

Cada producto software tiene una función y una forma. La función corresponde a los casos de uso y la forma a la arquitectura. Existe una interacción entre los casos de uso y la arquitectura; los casos de uso deben encajar en la arquitectura cuando se llevan a cabo y la arquitectura debe permitir el desarrollo de todos los casos de uso requeridos, actualmente y en el futuro. Esto provoca que tanto la arquitectura como los casos de uso deban evolucionar en paralelo durante todo el proceso de desarrollo de software.

c) Proceso iterativo e incremental

El desarrollo de un producto software supone un gran esfuerzo que puede durar desde varios meses hasta más de un año. Por lo tanto, una solución práctica para esto, es tener un proceso iterativo e incremental, en donde el trabajo se divida en partes más pequeñas o mini proyectos. Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto.

Igualmente, RUP lleva a cabo el desarrollo de un proyecto de software a través de un proceso iterativo e incremental. En el cual se plantea la implementación del proyecto a realizar en Iteraciones, con lo cual se pueden definir objetivos por cumplir en cada iteración y así poder ir completando todo el proyecto iteración por iteración; con lo cual se tienen varias ventajas, entre ellas se puede mencionar la de tener pequeños avances del proyecto, que son entregables al cliente el cual podría probar mientras se está desarrollando otra iteración del proyecto, con lo cual el proyecto va creciendo hasta completarlo en su totalidad.

2.1.18.2. Estructura de RUP

Mendoza (2008) indica que la estructura de RUP puede ser descrita a través de dos dimensiones o ejes:

Eje horizontal: representa el tiempo y es considerado el eje de los aspectos dinámicos del proceso. Esta dimensión muestra las características del ciclo de vida del proceso, expresado en términos de fases, iteraciones e hitos.

Eje vertical: representa los aspectos estáticos del proceso. Esta dimensión describe el proceso en términos de componentes de proceso, disciplinas, flujos de trabajo, actividades, artefactos y roles.

h. Estructura Dinámica de RUP

RUP se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo concluye con una versión del producto para los clientes. Cada ciclo consta de cuatro fases: Inicio, Elaboración, Construcción y Transición. Cada fase se subdivide a la vez en iteraciones, el número de iteraciones en cada fase es variable.

Fase de Inicio

En esta fase se define el modelo del negocio y el alcance del proyecto. Permite establecer una visión sobre el límite del sistema, el costo en recursos y tiempo. También, se evalúa la viabilidad del proyecto, sobre todo cuando está en juego una gran inversión de recursos; se estiman los riesgos asociados al proyecto y se identifican los casos de uso críticos del sistema y los escenarios básicos que definen la funcionalidad del sistema.

En esta fase inicial la actividad se concentra en los flujos trabajo de modelado del negocio y de requisitos, realizándose poco trabajo en los flujos

de trabajo de análisis y de diseño. Esta fase raramente realiza trabajo en los flujos de trabajo de implementación y de prueba.

Fase de elaboración

En esta fase se estudia a profundidad tanto la funcionalidad como el dominio del problema, se construye un prototipo de la arquitectura y se eliminan los mayores riesgos para lograr una culminación exitosa. El prototipo de la arquitectura, debe evolucionar en iteraciones sucesivas hasta convertirse en el sistema final. Este prototipo debe contener los casos de uso críticos identificados en la fase de inicio. También debe demostrarse que se han evitado los riesgos más graves. Con esta fase se busca definir, validar y cimentar la arquitectura, completar la visión y crear un plan fiable el cual puede variar con las iteraciones.

En la fase de elaboración, mientras continua una cierta actividad dedicada a completar los requisitos, los flujos de trabajo de análisis y de diseño reciben una gran parte de la actividad, ya que son la base de la creación de la arquitectura. Para alcanzar la arquitectura base sólida hay alguna actividad en los últimos flujos de trabajo (implementación y pruebas).

Fase de construcción

Lo que se busca en esta fase, es alcanzar la capacidad operacional del producto de forma incremental a través de sucesivas iteraciones. Durante esta fase se debe describir los requisitos restantes, refinar el diseño, e implementar, integrar y probar en su totalidad, todos los componentes, características y requisitos, obteniendo una versión aceptable del producto. En esta fase, por tanto, los distintos modelos del sistema van creciendo hasta completarse. La descripción de la arquitectura, sin embargo, no crece significativamente debido a que la mayor parte de esta arquitectura se definió durante la fase de elaboración.

Fase de transición

La finalidad de esta fase es poner el producto en manos de los usuarios finales, y entrenarlos en el manejo del mismo, y en general realizar tareas relacionadas con el ajuste, configuración, instalación y facilidad de uso del producto.

Una vez que el sistema se ha puesto en manos de los usuarios finales, a menudo aparecen cuestiones que requieren un desarrollo adicional para ajustar el sistema, corregir algunos problemas no detectados o finalizar algunas características que habían sido propuestas. Esta fase comienza normalmente con una versión beta del sistema, que luego será reemplazada por la versión definitiva del producto.

En esta fase de transición la actividad de los distintos flujos de trabajo depende de los resultados de las pruebas de aceptación y de las versiones beta del sistema. Por ejemplo, si las pruebas de las versiones beta descubren algún defecto en la implementación habrá una actividad considerable en los flujos de implementación y prueba.

Estructura estática de RUP

Está representada por cuatro elementos, que responden a las preguntas ¿Quién?, ¿Cómo?, ¿Qué? y ¿Cuándo? definidas en un proceso de desarrollo de software: los roles, responden a la pregunta ¿Quién?, las actividades responden a la pregunta ¿Cómo?, los productos, responden a la pregunta ¿Qué? y los flujos de trabajo de las disciplinas responden a la pregunta ¿Cuándo? En la Figura 9 se muestran algunos elementos de la estructura estática de RUP.

Roles

Un rol define el comportamiento y responsabilidades de un individuo, o de un grupo de individuos trabajando juntos como un equipo. Una persona

puede desempeñar diversos roles, así como un mismo rol puede ser representado por varias personas.

Las responsabilidades de un rol son tanto el llevar a cabo un conjunto de actividades como el ser el dueño de un conjunto de artefactos. RUP define grupos de roles, agrupados por participación en actividades relacionadas.

Actividades

Una actividad en concreto es una unidad de trabajo que se le asigna a una persona que desempeña un rol. Las actividades tienen un objetivo concreto, normalmente expresado en términos de crear o actualizar algún producto, tales como un modelo, una clase, un código fuente o un plan.

Artefactos

Los productos o artefactos son los resultados tangibles del proyecto, las cosas que se van creando, modificando y usando hasta obtener el producto final durante el proceso de desarrollo de software. Son las entradas y salidas de las actividades, realizadas por las personas que desempeñan roles, las cuales utilizan y van produciendo estos artefactos para tener guías.

Un artefacto puede ser cualquiera de los siguientes:

a. Un documento, como el documento de la arquitectura del software.
b. Un modelo, como el modelo de Casos de Uso o el modelo de diseño.
c. Un elemento del modelo, un elemento que pertenece a un modelo como una clase, un Caso de Uso o un subsistema.

De acuerdo a la filosofía de RUP (y de todo proceso iterativo e incremental), todos los artefactos son objeto de modificaciones a lo largo del proceso de desarrollo, con lo cual, sólo al término del proceso se podría tener una versión definitiva y completa de cada uno de ellos. Sin embargo, el resultado de cada iteración y los hitos del proyecto están enfocados a conseguir un cierto grado de completitud y estabilidad de los artefactos.

Flujos de trabajo (disciplinas)

Con simplemente la enumeración de roles, actividades y artefactos no se define un proceso, se necesita contar con una secuencia de actividades realizadas por los diferentes roles, así como la relación entre los mismos. Un flujo de trabajo es una relación de actividades que producen unos resultados observables.

Las disciplinas conllevan los flujos de trabajo, los cuales son una secuencia de pasos para la culminación de cada disciplina, estas disciplinas se dividen en dos grupos, como se apreció en la Figura 6, p. 42 las de proceso, y las de apoyo. Las de proceso son las necesarias para la realización de un proyecto de software; entre ellas se tienen: modelado del negocio, requerimientos, análisis y diseño, implementación, pruebas y despliegue. Las de apoyo son las que como su nombre lo indica sirven de apoyo a las de proceso y especifican otras características en la realización de un proyecto de software; entre estas se tienen: entorno, gestión del proyecto, gestión de configuración y cambios.

2.1.19. Arduino

Castro (2013) menciona que arduino es una plataforma electrónica de hardware libre basada en una placa con un microcontrolador. Con software y hardware flexibles y fáciles de utilizar, Arduino ha sido diseñado para adaptarse a las necesidades de todo tipo de público, desde aficionados, hasta expertos en robótica o equipos electrónicos.

Ante todo y sobre todo es un microcontrolador, es decir un ordenador completo integrado en un chip, con su CPU, memoria de programa, memoria de datos y circuitos para el control de periféricos.

Massimo (2012) indica que el microcontrolador necesita para su correcto funcionamiento, de algunos circuitos auxiliares y complementos tales como:

- La entrada de alimentación
- El oscilador de trabajo
- Circuito de RESET
- La conexión USB

- Los accesos a las líneas de entrada y salida, etc.

También consta de un simple, pero completo, entorno de desarrollo, que nos permite interactuar con la plataforma de manera muy sencilla. Se puede definir por tanto como una sencilla herramienta de contribución a la creación de prototipos, entornos, u objetos interactivos destinados a proyectos multidisciplinarios y multitecnología.

La placa Arduino está capacitada para incorporar hardware adicional, contiene una matriz de terminales en la que se puede añadir hardware de acuerdo al requerimiento del prototipo a desarrollar

a. Características

El Arduino Mega está basado en el microcontrolador ATMega2560. Tiene 54 pines de entradas/salidas digitales (14 de las cuales pueden ser utilizadas como salidas PWM), 16 entradas analogas, 4 UARTs (puertos serial por hardware), cristal oscilador de 16 Mhz, conexión USB, jack de alimentación, conector ICSP y botón de reset. Incorpora todo lo necesario para que el microcontrolador trabaje; simplementa conectalo a tu PC por medio de un cable USB o con una fuente de alimentación externa. El Arduino Mega es compatible con la mayoría de los shields diseñados para Arduino Duemilanove, diecimila o UNO.

Esta nueva versión de Arduino Mega 2560 adicionalmente a todas las características de su sucesor, el Arduino Mega ahora utiliza un microcontrolador ATMega8U2 en vez del chip FTDI. Esto permite mayores velocidades de transmisión por su puerto USB y no requiere drivers para Linux o MAC (archivo inf es necesario para Windows) además ahora cuenta con la capacidad de ser reconocido por el PC como un teclado, mouse, joystick, etc.

b. Alimentación de un arduino

El Arduino Mega puede ser alimentado vía la conexión USB o con una fuente de alimentación externa. El origen de la alimentación se selecciona

automáticamente. Las fuentes de alimentación externas (no USB) pueden ser tanto un transformador o una batería. El transformador se puede conectar usando un conector macho de 2.1mm con centro positivo en el conector hembra de la placa. Los cables de la batería puede conectarse a los pines Gnd y Vin en los conectores de alimentación (POWER) La placa puede trabajar con una alimentación externa de entre 6 a 20 voltios. Si el voltaje suministrado es inferior a 7V, el pin de 5V puede proporcionar menos de 5 Voltios y la placa puede volverse inestable; si se usan más de 12V los reguladores de voltaje se pueden sobrecalentar y dañar la placa. El rango recomendado es de 7 a 12 voltios.

Los pines de alimentación son los siguientes:

- VIN. La entrada de voltaje a la placa Arduino cando se está usando una fuente externa de alimentación (en opuesto a los 5 voltios de la conexión USB). Se puede proporcionar voltaje a través de este pin, o, si se está alimentando a través de la conexión de 2.1 mm acceder a ella a través de este pin.
- 5V. La fuente de voltaje estabilizado usado para alimentar el microcontrolador y otros componentes de la placa. Esta puede provenir de VIN a través de un regulador integrado en la placa, o proporcionada directamente por el USB u otra fuente estabilizada de 5V.
- 3V3. Una fuente de voltaje de 3.3 voltios generada por un regulador integrado en la placa. La corriente máxima soportada 50mA.
- GND. Pines de toma de tierra.

c. Memoria

El ATmega2560 tiene 256KB de memoria flash para almacenar código (8KB son usados para el arranque del sistema. El ATmega2560 tiene 8 KB de memoria SRAM y 4KB de EEPROM, a la cual se puede acceder para leer o escribir con la librería EEPROM.

d. Entradas y salidas

Cada uno de los 54 pines digitales en el Mega pueden utilizarse como entradas o como salidas usando las funciones pinMode(), digitalWrite(), y digitalRead(). Las E/S operan a 5 voltios. Cada pin puede proporcionar o recibir una intensidad maxima de 40mA y tiene una resistencia interna de pull-up (desconectada por defecto) de 20-50kOhms. Además, algunos pines tienen funciones especializadas:

Serie: 0 (RX) y 1 (TX),Serie 1: 19 (RX) y 18 (TX); Serie 2: 17 (RX) y 16 (TX); Serie 3: 15 (RX) y 14 (TX).Usados para recibir (RX) transmitir (TX) datos a través de puerto serie TTL. Los pines Serie: 0 (RX) y 1 (TX) estan conectados a los pines correspondientes del chip FTDI USB-to-TTL.

Interrupciones Externas: 2 (interrupción 0), 3 (interrupción 1), 18 (interrupción 5), 19 (interrupción 4), 20 (interrupción 3), y 21 (interrupción 2). Estos pines se pueden configurar para lanzar una interrupción en un valor LOW(0V), en flancos de subida o bajada (cambio de LOW a HIGH(5V) o viceversa), o en cambios de valor. Ver la función attachInterrupt() para más detalles.

PWM: de 0 a 13.Proporciona una salida PWM (Pulse Wave Modulation, modulación de onda por pulsos) de 8 bits de resolución (valores de 0 a 255) a traves de la función analogWrite().

SPI: 50 (SS), 51 (MOSI), 52 (MISO), 53 (SCK). Estos pines proporcionan comunicación SPI, usando lalibrería SPI.

LED: 13.Hay un LED integrado en la placa conectado al pin digital 13, cuando este pin tiene un valor HIGH(5V) el LED se enciende y cuando este tiene un valor LOW(0V) este se apaga.

El Mega tiene 16 entradas analógicas, y cada una de ellas proporciona una resolución de 10bits (1024 valores). Por defecto se mide desde 0V a 5V, aunque es posible cambiar la cota superior de este rango usando el pin AREF y la función analogReference().

I2C: 20 (SDA) y 21 (SCL). Soporte para el protocolo de comunicaciones I2C (TWI) usando la librería Wire.

AREF. Voltaje de referencia para la entradas analogicas. Usado por analogReference().

Reset. Suministrar un valor LOW (0V) para reiniciar el microcontrolador. Típicamente usado para añadir un botón de reset a los shields que no dejan acceso a este botón en la placa.

e. Comunicaciones

EL Arduino Mega 2560 facilita en varios aspectos la comunicación con la PC. El ATmega2560 proporciona cuatro puertos de comunicación vía serie UART TTL (5V). Un ATmega16U2bintegrado en la placa canaliza esta comunicación serie a través del puerto USB y los drivers (incluidos en el software de Arduino) proporcionan un puerto serie virtual en el ordenador. El software incluye un monitor de puerto serie que permite enviar y recibir información textual de la placa Arduino. Los LEDS RX y TX de la placa parpadearán cuando se detecte comunicación transmitida través de la conexión USB (no parpadearán si se usa la comunicación serie a través de los pines 0 y 1).

La librería Software Serial permite comunicación serie por cualquier par de pines digitales del Arduino Mega.

El ATmega2560 también soporta la comunicación I2C (TWI) y SPI. El software de Arduino incluye una librería Wire para simplificar el uso el bus I2C. Para el uso de la comunicación SPI, ver la hoja de especificaciones (datasheet) del ATmega2560 (Massimo, 2012).

f. Programación

El ATmega2560 en el Arduino Mega viene precargado con un gestor de arranque (bootloader) que permite cargar nuevo código sin necesidad de un programador por hardware externo. Se comunica utilizando el protocolo STK500 original (referencia, archivo de cabecera C).

También puede evitarse el gestor de arranque y programar directamente el microcontrolador a través del puerto ICSP (In Circuit Serial Programming).

g. Reinicio automático por software

En vez de necesitar reiniciar presionando físicamente el botón de reset antes de cargar, el Arduino Mega esta diseñado de manera que es posible reiniciar por software desde el ordenador donde esté conectado. Una de las líneas de control de flujo (DTR) del ATmega16U2 está conectada a la línea de reinicio del ATmega2560 a través de un condensador de 100 nanofaradios. Cuando la línea se pone a LOW(0V), la línea de reinicio también se pone a LOW el tiempo suficiente para reiniciar el chip. El software de Arduino utiliza esta característica para permitir cargar los sketches con solo apretar un botón del entorno. Dado que el gestor de arranque tiene un lapso de tiempo para ello, la activación del DTR y la carga del sketch se coordinan perfectamente.
Esta configuración tiene otras implicaciones. Cuando el Mega se conecta a un ordenador con Mac OS X o Linux, esto reinicia la placa cada vez que se realiza una conexión desde el software vía USB). El medio segundo aproximadamente posterior, el gestor de arranque se está ejecutando. A pesar de estar programado para ignorar datos mal formateados (ej. cualquier cosa que la carga de un programa nuevo) intercepta los primeros bytes que se envían a la placa justo después de que se abra la conexión. Si un sketch ejecutándose en la placa recibe algún tipo de configuración inicial u otro tipo de información al inicio del programa, debe asegurarse de que el software con el cual se comunica espera un segundo después de abrir la conexión antes de enviar los datos.

El Mega contiene una pista que puede ser cortada para deshabilitar el auto reset. Las terminaciones a cada lado pueden ser soldadas entre ellas para rehabilitarlo. Están etiquetadas con “RESET-EN”. También se puede deshabilitar el auto reset conectando una resistencia de 110 ohms desde el pin 5V al pin de reset.

h. Protección contra sobrecorrientes en USB

El Arduino Mega tiene un multifusible reinicializable que protege la conexión USB del PC de cortocircuitos y sobretensiones. Aparte de que la mayoría de ordenadores proporcionan su propia protección interna, el fusible proporciona una capa extra de protección. Si más de 500mA son detectados en el puerto USB, el fusible automáticamente corta la conexión hasta que la sobretensión desaparezca.

i. Características físicas y compatibilidad de Shields

La longitud y amplitud máxima de la placa Mega 2560 son de 4 y 2.1 pulgadas respectivamente, con el conector USB y la conexión de alimentación sobresaliendo de estas dimensiones. Tres agujeros para fijación con tornillos permiten colocar la placa en superficies y cajas. Tener en cuenta que la distancia entre los pines digitales 7 y 8 es 160 mil (0,16″), no es múltiplo de la separación de 100 mil entre los otros pines (Massimo, 2012).

2.2. Antecedentes

Darío (2009) en su investigación “Desarrollo de un software para el control de inventario de productos terminados para los departamentos de atención al cliente, la línea de producción “sector beta”, y despacho en una empresa alimentos”, donde plantea optimizar y reducir los tiempos y los errores, en la realización de las operaciones llevadas a cabo en los departamentos de: Atención al Cliente, Cuentas por Cobrar, Despacho, Línea de Producción, Romana, considerando los adelantos tecnológicos en el área de software, esto con el fin de obtener reportes más confiables y reales. Concluyó mencionando: 1. Se realizó un análisis detallado de las necesidades del sistema propuesto donde e especificaron todos los procesos requeridos para la elaboración de un reporte final “Orden de Despacho” y así lograr el objetivo inicialmente planteado. 2. Mediante la metodología de proceso unificado a través del modelado de casos de uso, se diseñaron clara y

eficientemente los módulos que conforman el sistema, lo que permitió desarrollar una arquitectura solida del mismo. 3. Se logró diseñar y desarrollar a través de los diferentes diagramas de UML un software, robusto, amigable y de fácil manipulación por el usuario final; y de esta forma se reduce el margen de error por parte de los usuarios del sistema. 4. Mediante la aplicación de esta herramienta, se logran disminuir los tiempos de respuesta, en cuanto a elaboración de las órdenes de pedido y despacho, en función al control de despacho y de inventario.

Martínez (2011) en su trabajo de investigación “Desarrollo de un software para la automatización de los procesos administrativos de la sección de almacén del núcleo monagas de la universidad de oriente” tuvo como propósito principal el desarrollar un sistema para automatizar los procesos administrativos de la sección de almacén del Núcleo Monagas de la Universidad de Oriente. Para ello fue necesario estudiar el funcionamiento actual de dicha sección, y determinar la problemática que presentaba en la prestación de sus servicios; para luego, definir los requerimientos de información del sistema en base a dicha problemática y a las necesidades del personal que labora en el departamento en cuestión; procediéndose después a diseñar una arquitectura sólida que cumpliera con todos los requerimientos establecidos, hasta finalmente obtener un prototipo inicial de la aplicación, de acuerdo a esa arquitectura diseñada. De igual manera, pudo concluir que con el desarrollo y futura implantación del sistema se agilizarán los procesos administrativos llevados a cabo en dicha sección, tales como la generación de reportes de productos existentes en el almacén lo que traerá consigo un ahorro significativo del tiempo de respuesta y una carga de trabajo mucho menor para los trabajadores que laboran en la sección de almacén

Loya (2010) en su investigación “Diseño e implementación de un programa de software para el dimensionamiento de un secador de doble tambor” menciona que su Proyecto de Titulación abordó el desarrollo de dos programas SETA I Y SETA II, escritos en Visual Basic 6.0, y permiten obtener las dimensiones de un secador continuo de doble tambor. Para acopiar datos

para el desarrollo de los programas, durante el desarrollo del proyecto se realizaron pruebas experimentales de secado en las que se alimentó pasta de banano al secador en lotes de 0,800 kg (modalidad semi-batch) para establecer la influencia de la presión del vapor saturado, de la velocidad de rotación de los tambores, y de la separación entre ellos; sobre la humedad final, el flujo másico y la velocidad de evaporación del agua. Para preparar la pasta, se caracterizó el banano respecto de su color externo, humedad inicial y sólidos solubles. Además del estudio de la influencia entre las variables citadas, las pruebas de deshidratación permitieron estimar los coeficientes de transferencia de calor, la velocidad del aire circundante, la humedad final, y los flujos másicos que se utilizaron en los programas. Para comprobar la idoneidad del programa SETA II, éste se ejecutó utilizando flujos másicos de agua evaporada obtenidos de secadores comerciales industriales de doble tambor, comparándose los resultados así obtenidos por los programas con las dimensiones de secadores industriales mediante la variación porcentual relativa, pudiéndose concluir que el programa obtiene aproximaciones razonables a los secadores industriales.

Pilligua (2006) en su investigación “Diseño de un software para calcular cámaras frigoríficas”, en este proyecto se desarrolla un software para seleccionar el espesor de las paredes de la cámara y el equipo de refrigeración, en base a las dimensiones dadas de la cámara, la descripción del producto, las dimensiones de la puerta, número de personas trabajando dentro de la cámara y el tiempo que estén dentro de esta, y las temperaturas del medio donde esta se encuentra ubicada. Empezando con la descripción de los factores de influencia de los alimentos y las maneras que estos pueden ser conservados. Continuamos con la descripción de los refrigerantes, tipos de aislantes y con la descripción de los diferentes accesorios y equipos de refrigeración utilizados en la industria. Luego empezamos a realizar la parte de los cálculos y seleccionamiento del aislante para las paredes de la cámara frigorífica, del sistema de refrigeración, así como la selección de sus accesorios. Asimismo concluye que se creó un Software para diseñar de una manera rápida y sencilla cámaras frigoríficas, con su respectivo sistema de

refrigeración, estas pueden ser para cualquier producto producido en nuestro País.

Aleman (2015) en su trabajo de investigación "Determinación de parámetros adecuados en la elaboración de un néctar tropical mixto de mango (Manguifera indica L) con ciruela (Spondias purpurea L)", tuvo por objetivo Determinar los parámetros adecuados para la elaboración de un néctar tropical mixto de mango (Manguifera indica L) con ciruela (Spondias purpurea L), llegando a obtener como resultado de las investigaciones se determinó que la proporción adecuada que permite un equilibrio entre las características sensoriales del mango y ciruela es de 70 partes de pulpa de mango por 30 partes de pulpa de ciruela en la elaboración del néctar tropical mixto; la dilución adecuada que permite resaltar mejor las características sensoriales de la pulpa mixta de mango y ciruela es de una parte de pulpa por cuatro partes de agua; las características fisicoquímicas recomendadas por la NTP N° 203.110-2009 para el néctar más aceptado encontrándose un brix igual a 15, % de acidez igual a 0,17, pH igual a 4,30 y vitamina C igual a 9,26 mg/100g y evaluó la vida de anaquel del néctar elaborado mantenido a condiciones ambientales encontrándose que hasta los 120 días de elaborado los parámetros fisicoquímicos y microbiológicos se encontraban dentro de los límites de la NTP N° 203.110-20 y NTS N° 071-MINSA/DIGESA-V.01

2.3. Hipótesis

2.3.1. Hipótesis general

- Si diseñamos y desarrollamos un software de calidad (CAM) entonces su aplicación influirá en la eficiencia del proceso de elaboración de néctar.

2.3.2. Hipótesis específicas

- Si desarrollamos el software con estándares de calidad en la codificación, entonces su aplicación influirá eficientemente en la formulación de la elaboración de néctar.
- El desarrollo del software influirá en la eficiencia del control del proceso de elaboración de néctar a mediante la sistematización de la información.
- Las características fisicoquímicas y sensoriales del néctar elaborado, al usar el software como herramienta en el proceso de elaboración serán aceptables por los consumidores.

2.4. Variables

2.4.1. Variables independientes

- CAM - Manufactura asistida por computadora

2.4.2. Variables dependientes

- Formulación
- Control de proceso
- Características fisicoquímicas
- Características sensoriales

2.4.3. Operacionalización de variables

Tabla 1. *Operacionalización de variables*

Variables		Definición de variables	Dimensiones	Indicadores	Instrumento	Ítems
VARIABLES INDEPENDIENTES	CAM - Manufactura asistida por computadora	Uso de aplicaciones de software para controlar sistemas de producción automatizados.	Software	- Programación - Codificación - Arquitectura del software - Visual Basic .NET - NetFrameworks 4.54.5	Ordenador Software de programación	¿El lenguaje y software de programación serán pertinentes y óptimos para la programación?
VARIABLES DEPENDIENTES	Formulación	Conjunto de reglas preestablecidas que han de seguirse de manera ordenada para expresar un compuesto mediante su fórmula.	Dosificación	- Fruta (g.) - Agua (g.) - Azúcar (g.) - Espesante (g.) - Conservante (g.) - Corrector de pH (g.)	Software de programación y aplicación	¿La formulación realizada con la ayuda del software, será eficiente para realizar el proceso?
	Control de proceso	Conjunto de operaciones a que se somete una materia prima para elaborarla o transformarla.	Control	- Temperatura (ºC) - Tiempo (min)	Software de programación y aplicación	¿El software influirá eficientemente en el control del proceso?
	Características fisicoquímicas	Propiedades del producto que podemos observar y medir, cuando este haya sufrido cambios en su composición.	Evaluación fisicoquímica	- ºBrix - pH	Medición	¿Las características fisicoquímicas del néctar elaborado con la ayuda del software serán aceptables?
	Características sensoriales	Cualidades del producto, que son determinadas por el olor, sabor, color, el brillo, la textura y el aspecto general; varían según la especie.	Evaluación sensorial	- Color - Olor - Sabor	Encuesta	¿Las características sensoriales del néctar elaborado con la ayuda del software serán aceptables?

Fuente y elaboración: Autor (2018).

III. MATERIALES Y MÉTODOS

3.1. Tipo y nivel de investigación

- Tipo de investigación: aplicada
- Nivel de investigación: experimental

3.2. Lugar de ejecución

El presente proyecto de investigación se realizó en los ambientes del Centro de Innovación y Emprendimiento Agroindustrial, y la parte experimental en la planta de procesos alimentarios de la Escuela Académico Profesional de Ingeniería Agroindustrial de la Universidad Nacional Hermilio Valdizán en el departamento de Huánuco.

3.3. Población, muestra y unidad de análisis

- Población: 30 estudiantes y profesionales de la escuela profesional de ingeniería agroindustrial.
- Muestra: 30 estudiantes y profesionales de la escuela profesional de ingeniería agroindustrial.
- Unidad de análisis: néctar de cocona.

3.4. Tratamiento en estudio

Para la presente investigación existe un solo tratamiento respecto al néctar elaborado bajo la asistencia del software y otro tratamiento como muestra de comparación que es un néctar elaborado sin la asistencia del software.

Tabla 2. *Tratamientos en estudio*

Tratamientos en estudio	Características	Análisis estadístico
Tratamiento 1	Con asistencia del software	Ji cuadrado y Prueba T
Tratamiento 2	Sin asistencia del software	

3.5. Prueba de hipótesis

Hipótesis nula

H_0: el software (CAM) no influirá en el proceso de elaboración del néctar.

H_0: $\mu_1 = \mu_2$

Hipótesis de investigación

H_1: el software (CAM) influirá en el proceso de elaboración del néctar.

H_1: $\mu_1 \neq \mu_2$

3.5.1. Diseño de la investigación

a. Prueba Chi-cuadrado: aceptación del producto

Se ha utilizado esta prueba para observar la aceptación del producto, para ajustarse a un modelo de chi-cuadrado se utilizó como el más adecuado la prueba de bondad de ajuste.

- **Primer paso:** formulación de las hipótesis nula y alternativa
  - **H0:** el software desarrollado **no** tendrá aprobación de los usuarios.
  - H_0: $\mu_1 = \mu_2$
  - **H1:** el software desarrollado **si** tendrá aprobación de los usuarios.
  - H_1: $\mu_1 \neq \mu_2$
- **Segundo paso:** identificación, si es cola derecha o izquierda.
- **Tercer paso:** nivel de significancia.
- **Cuarto paso:** estadístico de la prueba.
- **Quinto paso:** se realiza el esquema de la prueba.
- **Sexto paso:** cálculo del estadístico de la prueba.

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

$con\ [k-1]\ Grados\ de\ libertad$

- **Séptimo paso:** decisión
  Se analizó si el software desarrollado tendrá o no aprobación de los usuarios.

b. Prueba Chi-cuadrado de asociación u homogeneidad: influencia de la programación en la elaboración de productos

Se utilizó esta prueba de asociación u homogeneidad para determinar si el valor observado de una variable depende del valor observado de otra variable, o para determinar si una variable está asociada a otra variable. En este caso se compara la programación en el desarrollo de software utilizando calidad de información con la elaboración del producto con dicha información. Comparar si existe relación o asociación entre ambos es decir si al utilizar la información de calidad estructurado en el software conlleva a obtener productos de buena calidad.

- **Primer paso:** formulación de las hipótesis nula y alternativa
  H0: el desarrollo del software con calidad de información no influirá en el proceso de elaboración del néctar.
  $H_0: \mu_1 = \mu_2$
  H1: el desarrollo del software con calidad de información influirá en el proceso de elaboración del néctar.
  $H_1: \mu_1 \neq \mu_2$
- **Segundo paso:** identificación, si es cola derecha o izquierda
- **Tercer paso:** nivel de significancia.
- **Cuarto paso:** estadístico de la prueba.
- **Quinto paso:** se realiza el esquema de la prueba
- **Sexto paso:** cálculo del estadístico de la prueba

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

$con\ [n-1][k-1]\ Grados\ de\ libertad$

- **Séptimo paso:** decisión

  Se analizó si el desarrollo del software con calidad de información influye o no en el proceso de elaboración del néctar.

c. Prueba T muestras independientes: comparar el néctar elaborado con la utilización del software con néctar elaborado sin la utilización de software.

Prueba t para muestras independientes compara las medias de dos grupos de casos. En ese caso comparar el néctar elaborado con la utilización del software con néctar elaborado sin la utilización de software.

- **Primer paso:** formulación de las hipótesis nula y alternativa

  H0: no existe diferencia significativa entre la elaboración del néctar con la asistencia del software utilizando información de calidad y la elaboración del néctar sin la utilización de software si ninguna información de calidad.

  $H_0: \mu_1 = \mu_2$

  H1: existe diferencia significativa entre la elaboración del néctar con la asistencia del software utilizando información de calidad y la elaboración del néctar sin la utilización de software si ninguna información de calidad.

  $H_1: \mu_1 \neq \mu_2$

- **Segundo paso:** nivel de significancia.

- **Tercer paso:** estadístico de la prueba.

- **Cuarto paso: prueba de levene** prueba de igualdad de varianza.
  a. P-valor => α aceptar h0 = las varianzas son iguales
  b. P-valor < α aceptar h1 = existe diferencias significativas entre las varianzas.

- **Quinto paso:** decisión estadística

3.5.2. Datos a registrar

Los datos a registrarse son los siguientes:

- Porcentaje de encuestados que utilizan algún software en su trabajo o estudio.
- Tipo de software que utilizan con más frecuencia.
- Aceptación de la arquitectura del software.
- Aceptación de la eficiencia del software.
- Características organolépticas del producto.
- Porcentaje de aceptación del software en la importancia del control del proceso.
- Porcentaje dispuesto a adquirir el software.
- Formulación de insumos para elaboración néctar.
- Diagrama de flujo para la elaboración de nácar.

3.5.3. Técnicas e instrumentos de recolección y procesamiento de datos de la información

Para la obtención y registro de datos se utilizaran formatos elaborados acorde al estudio, memorias USB para el almacenamiento de datos, cuadernos de apuntes, lápices, marcadores, etc.

a. Técnica de investigación documental bibliográfica

- El análisis documental: todo lo referente al desarrollo de software y a las normas que rigen para el cumplimiento de algunos criterios a seguir, nos permitirá realizar eficientemente la investigación.
- El fichaje: permitirá registrar aspectos esenciales de los materiales consultados, estos servirán para redactar el marco teórico.

b. Instrumento de recolección de información en laboratorio

Hojas de registro, cuaderno de apuntes, cámara fotográfica.

c. Procesamiento y presentación de resultados

Los datos obtenidos fueron ordenados y procesados con la ayuda del ordenador (PC), utilizando los softwares de Word, Excel y del programa

estadísticos de IBM SPSS Statistics 22, estos datos serán presentados de manera textual y gráfica.

3.6. Materiales y equipos

3.6.1. Materiales de laboratorio de programación

- Estante para PC
- Silla

3.6.2. Materiales de escritorio

- Lapiceros
- Cuadernos
- Plumones
- DVD’S
- Hojas de bond
- USB’S

3.6.3. Equipos de programación

- Computadora
- Placa Arduino
- Sensor de temperatura
- Impresora
- Resistencia
- Parlante
- Teléfono

3.6.4. Insumos para elaboración del néctar

- Fruta
- Azúcar
- Estabilizante
- Conservante
- Regulador de pH.

3.6.5. Materiales para la elaboración del néctar

- Recipiente
- Jarras
- Coladores
- Tablas de picar
- Cuchillo de cocina
- Cucharas d medida
- Tamiz
- Paletas
- Refractómetro
- pH-Metro
- Termómetro

3.6.6. Equipos para la elaboración del néctar

- Pulpeadora, marca Corinper, 80 kilogramos, Peruana.
- Licuadora, marca Corinper, 2O litros, Peruana.
- Balanza, PCE-EP 1500, 50 Kilos, Española.

3.7. Conducción de la investigación

Tabla 3. *Etapas de la conducción de la investigación*

Etapas	Fase	Metodología	Objetivo	Actividades
I	Análisis de requisitos	RUP	Identificar el proceso y la problemática existente en el proceso de elaboración de néctar.	Entrevistas no estructuradas. Observación directa. Revisión Documental. Diseño del Plan de desarrollo de Software.
II	Diseño de la arquitectura	RUP	Determinar los requisitos básicos del diseño del Software. Diseñar la Arquitectura óptima del software para la elaboración de néctar.	Definición de la arquitectura del software. Modelo de Casos de Uso del Sistema. Especificación de casos de uso del sistema.
III	Diseño y desarrollo del software para elaboración de néctar.	RUP	Diseñar y desarrollar el Software usando la metodología basada en objetos para la eficiencia en la elaboración de néctar.	Codificación. Integración. Pruebas unitarias y de integración. Modelo de despliegue. Aplicación de pruebas finales al software y corrección de errores.
IV	Diseño y desarrollo del software para control del proceso.	RUP	Desarrollar el Software usando la metodología basada en objetos para el control del proceso.	Codificación. Integración. Pruebas unitarias y de integración. Modelo de despliegue. Aplicación de pruebas finales al software y corrección de errores.
V	Desarrollo del proceso de elaboración de néctar	Procesos alimentarios	Elaborar el néctar con la asistencia y sin la asistencia del software.	Elaboración de néctar. Aplicación de uso del software desarrollado.
VI	Determinación de las características fisicoquímicas y sensoriales del néctar y análisis estadístico	Metodología de la investigación (Estadística)	Realizar el análisis estadístico para el contraste de hipótesis.	Encuestas. Recolección de datos. Procesamiento de datos. Presentación de datos.

3.7.1. Etapa I.- Análisis de requisitos

En esta primera etapa se analizó la problemática actual que se presenta en la elaboración de néctar.

Captura, elicitación, especificación y análisis de requisitos (ERS)

Primeramente lo que se realizó es la captura de datos, todo lo relacionado a procesos, estos datos han sido estrictamente analizados para poder ser utilizados para la formulación de algoritmos, especificando las funcionalidades que deberán tener y se analizó los requisitos que deberá cumplir el futuro programa.

Estos datos y requisitos son enfocados al campo agroindustrial, la cual se produjo de las siguientes interrogantes:

a) La Funcionalidad.
¿Qué se supone va hacer el software?

b) Las interfaces Externas.
¿Cómo el software actúa recíprocamente con las personas, el hardware de los sistemas, otro hardware, y otro software?

c) La Actuación.
¿Cuál es la velocidad, la disponibilidad, tiempo de la contestación, tiempo de la recuperación de varias funciones del software, etc.?

d) Los Atributos.
¿Qué portabilidad tiene, exactitud, el mantenimiento, la seguridad, las consideraciones, etc.?

e) Las restricciones del diseño que impusieron en una aplicación.
¿Hay algún requerimiento Standard, idioma de aplicación, las políticas para la integridad del banco de datos, los límites de los recursos, operando en qué ambiente(s) etc.?

Las bondades de las características, del programa desarrollado, como de su entorno, parámetros no funcionales y arquitectura dependen enormemente de lo bien lograda que esté esta etapa. Esta es, probablemente, la de mayor importancia y una de las fases más difíciles de lograr certeramente, pues no es automatizable, no es muy técnica y depende en gran medida de la habilidad y experiencia de nuestras manos.

El conjunto de artefactos de RUP que se elaboraron en esta primera etapa, y que fueron utilizados durante todo el proyecto son los siguientes:

a. Plan de iteración.
b. Documento visión, donde están definidas las características del producto a desarrollar.
c. Especificación de casos de uso.

3.7.2. Etapa II.- Diseño de la arquitectura

Este proceso es muy importante, ya que de aquí partió nuestra proyección de nuestro producto hacia el usuario final, se trata básicamente de la arquitectura del software porque aquí se define el diseño del software, el cual abarca toda la interfaz gráfica del producto desde el momento que se ejecuta el “Setup” hasta realizar el ultimo clic en la pestaña de “cerrar ventana”, entre estas características son los siguientes:

- Arquitectura del software: en esta fase se planteó todo el diseño del software, desde el tipo y combinación de colores, tipo y tamaño de letra, diseño de las presentaciones, herramientas principales y auxiliares, cuadros de diálogos, aplicativos adicionales, entro otros; para lo cual se ha utilizado el cuadro de herramientas del programa, ya que la versión utilizada cuenta con una gama de herramientas actualizadas y automatizadas.

- Instalador (Zetup): en este proceso se estableció el tipo de instalación, estableciendo las mínimas características, esto se realizó al final de la programación, para realizar dicho proceso se tiene que descargar a

través de internet de la página web de Microsoft Support (Soporte de Microsoft) el aplicativo de Install Shield e integrarlo al programa de Visual Estudio.

3.7.3. Etapa III.- Diseño y desarrollo del software para elaboración de néctar

Codificación

Se realizó la codificación en base a las preguntas realizadas en la primera etapa de especificación y análisis de requisitos, esta programación se realizó en el leguaje mencionado, para lo cual se utilizaron varias normas reglamentados por la Ingeniería del Software, en esta etapa se realizó las siguientes actividades:

A. Primeramente se realizó un análisis de datos que van a ser procesados, con un estricto control de requisitos.

B. Seguidamente estos datos se llevó a código fuente, para lo cual se crearon formularios y en estos hacemos la formulación de algoritmos de cada uno de estos datos empezando por la arquitectura es decir el diseño del software, añadido a esto se programara lo más importante, es decir la funcionalidad de cada dato procesado, empezando por los procesos agroindustriales seguidamente por los aplicativos complementarios.

C. Asimismo después de programar se realizó la prueba de depuración, cabe mencionar que este proceso también se realizó mientras se programa, esta tarea se lleva a cabo para observar la cantidad de errores, y de esa manera poder corregirlo antes de que estos errores sean descubiertos por el usuario, el control de estos errores la realizamos de dos maneras: la primera a través de la programación, es decir volviendo a controlarlos mediante códigos, y la segunda cambiando o eliminando los factores que generan las

falencias. Igualmente esta etapa se realizó gracias al depurador integrado dentro del programa utilizado (IDE).

D. Al realizar la depuración, se procedió con la compilación para dar por finalizado la programación considerando que esté todo correcto, la compilación se realiza al final ya que esta compilación es realizado de manera general, dando por terminado la programación de los procesos agroindustriales como también a los aplicativos incorporados. Cabe mencionar que esta depuración se realiza con el mismo programa utilizado, ya que cuenta con un robusto depurador de códigos (IDE).

Integración y prueba del sistema

Se tuvo en consideración los siguientes aspectos:

a. Pruebas unitarias: se realizó una prueba individual de cada parte del software, es decir es probado proceso por proceso, como ya se ha mencionado esta pruebas se realiza con la finalidad de encontrar y corregir errores antes de ser instalado. Esta prueba se realizó con el ejecutador del programa y también se usa en momentos los compiladores y depuradores de la IDE del programa.

b. Pruebas de integración: en aquí se analizó de manera general de todo el proyecto en si este proceso es un control de calidad tanto de la arquitectura como de la codificación realizada. Esta prueba se realizó con el ejecutador y con el depurador del IDE del programa usado.

Instalación y paso a producción

Asimismo se procedió con la instalación, es decir con pasar todo lo programado a la computadora para que sea usado por los usuarios finales; por lo tanto esta es nuestra etapa final de desarrollo; de allí en adelante será usado para el fin que fue diseñado.

La instalación del software es el proceso por el cual los programas desarrollados son transferidos apropiadamente al computador destino, inicializados, y, eventualmente, configurados; todo ello con el propósito de ser ya utilizados por el usuario final. Constituye la etapa final en el desarrollo propiamente dicho del software. Luego de ésta el producto entró en la fase de funcionamiento y producción, para el que fue diseñado.

Operación y mantenimiento

En esta etapa lo que se realizo es básicamente ver la funcionalidad, la calidad, la eficiencia y también los posibles errores que no la hemos podido encontrar durante las etapas anteriores, vale mencionar que esta etapa es a tiempo indefinido o hasta el tiempo en que tendremos que brindar el soporte técnico a nuestros usuarios.

Control de proceso

Se realizó la integración del proceso de automatización a través del sistema de programación del entorno de Visual Basic con Arduino, se comenzó con el desarrollo de la aplicación cargando los driver necesarios para la conexión del dispositivo periférico, asimismo se realizó la integración con la placa Arduino y el sensor d temperatura con el control del tiempo y temperatura.

3.7.4. Etapa IV.- Diseño y desarrollo del software para control del proceso

Integración y arquitectura grafica de control de proceso

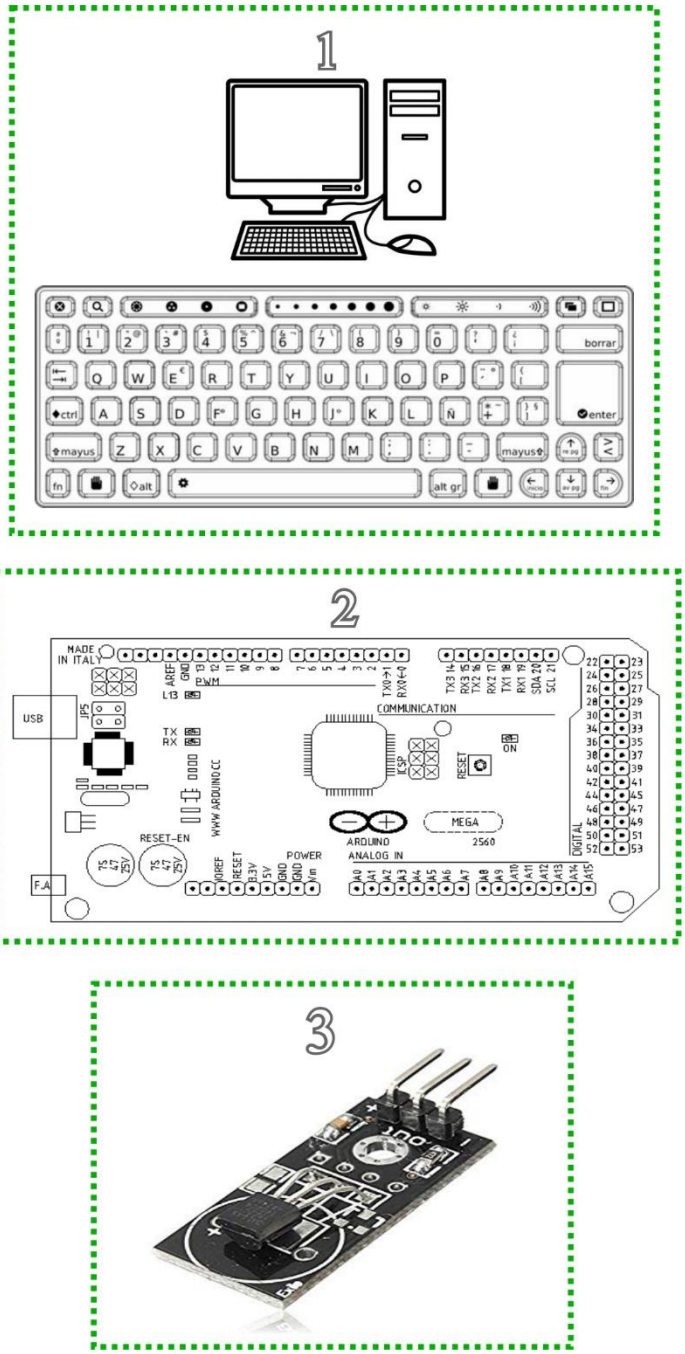

Figura 21. Diagrama de control de proceso.

a. Aplicación para control de proceso

Se desarrolló la aplicación en el programa de visual estudio, con la plataforma de Microsoft .Net, con una integración de librerías para el

lenguaje de Visual Basic, así mismo la codificación de los algoritmos para la lectura, procesamiento impresión de datos en tiempo real del dispositivo conectado con el sensor de temperatura.

b. Integración con Arduino

Visual Basic y puerto serie

La interfaz serie asíncrona es el principal dispositivo de comunicación de sistema a sistema. Asíncrono significa que no hay presente una señal de sincronización o de reloj. Cada carácter está enmarcado entre señales de inicio y parada. Un solo bit 0, denominado bit de inicio, precede a cada carácter para indicar al sistema que los siguientes 8 bits constituyen un byte de datos. Uno o dos bits en alto siguen al carácter para señalar que dicho carácter ha sido enviado.

Trama en una transmisión asíncrona:

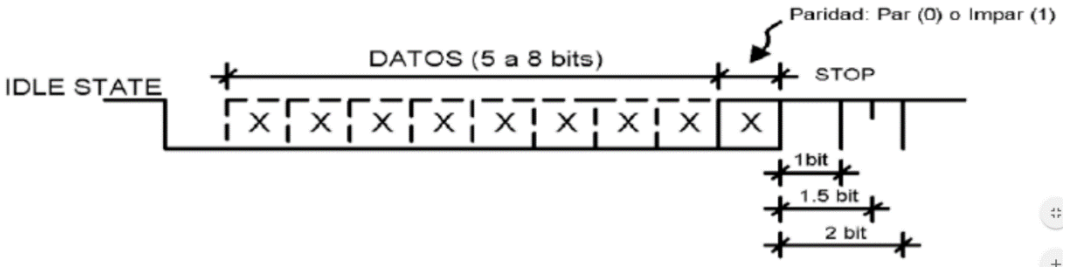

Figura 22. Trama Asíncrona.

El puerto serie en una computadora está compuesto por varias entradas/salidas. Dispuestas en un conector del tipo DB9 o DB25, tal como se muestra en la siguiente figura:

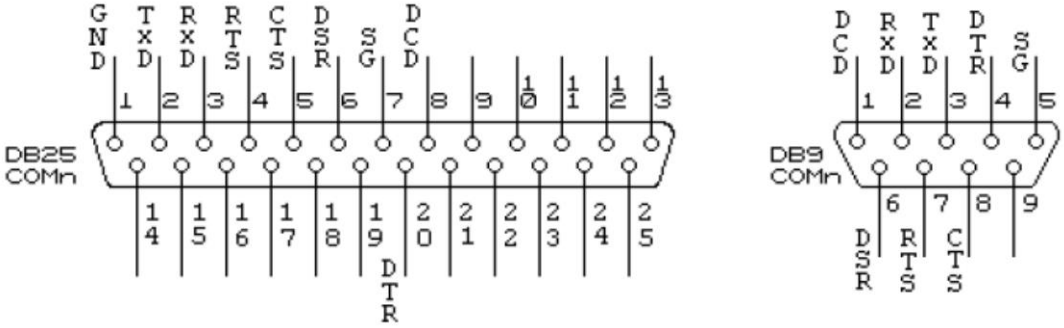

Figura 23. Conector DB25, b) Conector DB9.

La distribución de las señales en cada uno de sus pines es la siguiente:

Tabla 4. *Distribución de las señales*

Señal	**Descripción**
GND o SG	0 Voltios
TxD	Salida para transmisión de datos
RxD	Entrada para recepción de datos
RTS	(Request To Send) Salida que indica una petición de envío
CTS	(Clear To Send) Dispuesto para enviar, entrada por donde le indica el otro dispositivo que ya puede enviar los datos.
DSR	(Data Set Ready) Dispositivo de datos preparado, entrada por donde le indica el otro dispositivo que ya está listo.
DCD o CD	Entrada para la detección de portadora
DTR	(Data Terminal Ready) Salida, terminal de datos listo

Microsoft comm control

Es el control que permite la comunicación de una aplicación hecha en Visual Basic con el puerto serie. No está en la caja de herramientas por defecto, debe introducirse mediante el menú Proyecto y luego Componentes. En el formulario solamente se ve en tiempo de diseño.

Para habilitar la herramienta del puerto serie en Visual Basic se realizó lo siguiente: Seleccionando la opción Proyecto>>Componentes, al aparecer el listado de componentes seleccionar: Microsoft comm control, clic en aceptar, lo que provocará que la barra de herramientas cambie como se muestra en la siguiente figura:

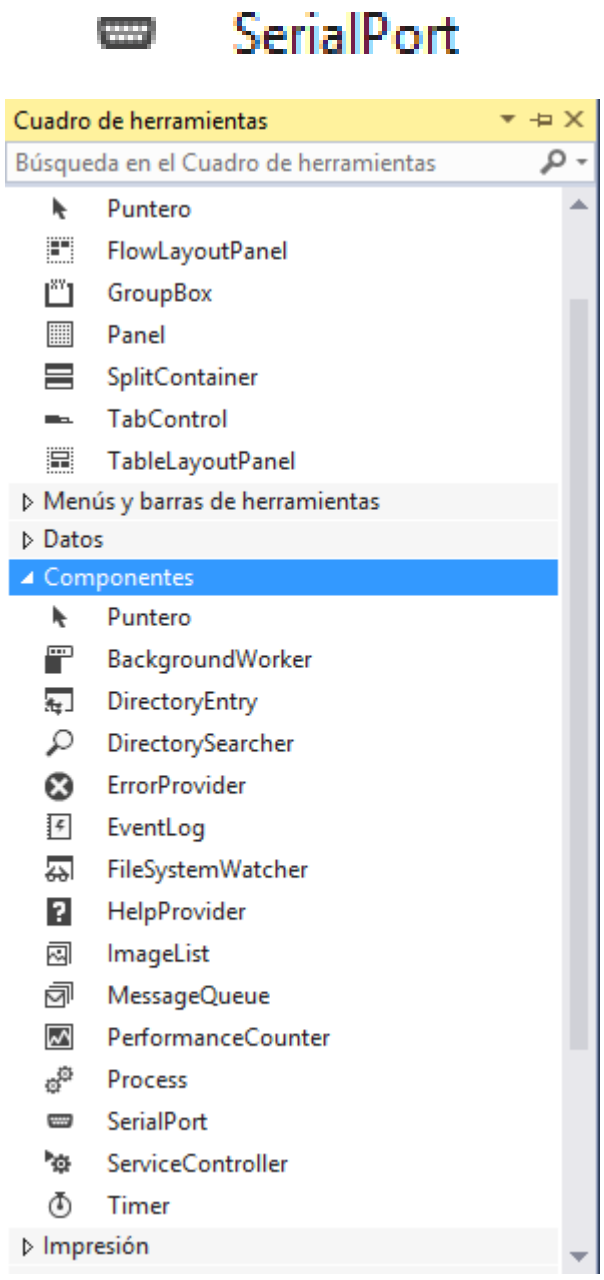

Figura 24. MSCOMM Control.

Propiedades

Entre las propiedades del MSCOMM Control hay algunas que pueden establecerse en tiempo de diseño o en tiempo de ejecución, y otras que solamente se pueden establecer o consultar en tiempo de ejecución.

A continuación se enuncian algunas propiedades del control

Tabla 5. *Propiedades de control*

Componente	**Propiedades**
CommPort SerialPort	Indica el número del puerto serie a utilizar, admite valores del 1 al 255, Generalmente las PC tienen dos puertos seriales: Com1 y Com2, si se le coloca un número de puerto inexistente dará error
Settings	Indica 4 parámetros en orden: velocidad, paridad, número de bits y bits de parada que se usarán en la comunicación
InBufferSize	Mediante esta propiedad se establece el tamaño del buffer de entrada. Puede conocerse el número de caracteres presentes en el buffer de entrada consultando el valor de la propiedad InBufferCount
OutBufferSize	Mediante esta propiedad se controla el tamaño del buffer de salida. Puede conocerse el número de caracteres presentes en el buffer de salida (los que aún están transmitirse), consultando el valor de la propiedad OutBufferCount
PortOpen	Abre el puerto de comunicación. Puede tener los valores True (para abrirlo) y False (para cerrarlo)
InBufferCount	Permite averiguar cuántos caracteres se tienen en el buffer de entrada
OutBufferCount	Permite conocer cuántos caracteres quedan por transmitir en el buffer de salida
Output	Envía caracteres al buffer de salida
Input	Lee el buffer de recepción

El control Microsoft Comm Control 6.0 tiene por defecto el nombre MSComm1. Para establecer o consultar una propiedad se debe utilizar la siguiente sintaxis MSComm1.Propiedad

Por ejemplo, para abrir el puerto se coloca MSComm1.Portopen=True

Para leer el puerto Buffer=MSComm1.Input

Para escribir en el puerto MSComm1.Output=dato

De igual forma con las demás propiedades

Manejo de periféricos analógicos mediante arduino

La plataforma Arduino, basada en microcontrolador AVR, hereda la capacidad intrínseca de dicho microcontrolador para manejar periféricos analógicos mediante técnicas de conversión específicas. Dichas técnicas consisten en la conversión analógica a digital (ADC) y modulación por ancho de pulso (PWM). Cada una permite interactuar con periféricos analógicos que proveen información de entrada (ADC) o que aceptan información de salida (PWM).

Conversión analógica a digital

El microcontrolador cuenta con un ADC integrado en el mismo chip, que permite tomar lecturas analógicas de voltaje de un dispositivo que genere tensiones entre 0 y 5V. La configuración de dicho módulo es automática, bastando solamente con invocar la función de lectura de datos, la cual se detalla a continuación: **analogRead (número_pin):** realiza la lectura sobre el pin analógico que se le especifique (Sólo se permiten los pines A0 al A5). El valor que devuelve es un número entero proporcional al voltaje de entrada, de tal forma que si se introducen 0V devuelve un valor de 0 y si se introducen 5V devuelve un valor de 1023, con todos los voltajes intermedios se producen valores distribuidos linealmente a lo largo de este intervalo.

Nótese que no es necesario configurar previamente un pin analógico para poderlo usar con esta función.

Advertencia: el rango válido para los voltajes de entrada es solamente de 0 a 5V. Si introduce voltajes negativos, o bien, voltajes positivos arriba de 5V, podrá dañar de manera permanente el dispositivo.

Modulación por ancho de pulso (PWM)

El Microcontrolador AVR cuenta también con dispositivos temporizadores que permiten generar señales externas, las cuales emplean la modulación por ancho de pulso para manejar cargas que sean compatibles con este esquema. Vale aclarar que la técnica en sí, es completamente diferente de la conversión digital a analógica (DAC), puesto que no se generan tensiones analógicas constantes (por ejemplo entre 0 y 5V) en ningún momento, sino que más bien la carga es encendida y apagada rápidamente, produciendo un efecto equivalente a proveer un nivel de potencia intermedio, que puede variar entre completamente apagado y completamente encendido. Además, de manera similar al ADC, los módulos temporizadores son inicializados de manera automática, bastando con invocar la función que se detalla a continuación:

AnalogWrite (número_pin, ciclo_de_trabajo): genera una señal PWM sobre un pin digital que tenga esa capacidad (denotado con ~). El rango válido para el ciclo de trabajo va desde 0 (completamente apagado) hasta 255 (completamente encendido).

No todas las cargas son compatibles con la modulación PWM, por lo que se recomienda que se informe antes de emplearla con algún dispositivo en particular. La técnica funciona bien con algunas cargas como LEDs y motores DC pequeños, mas no con cargas como relés o motores DC sin escobillas (como los ventiladores para chasís de computadora).

Lectura de entradas analógicas

- Procediendo a ensamblar el circuito junto con el Arduino. Se realizó la conexión directamente del sensor a la placa Arduino con la interpuesta de una resistencia.

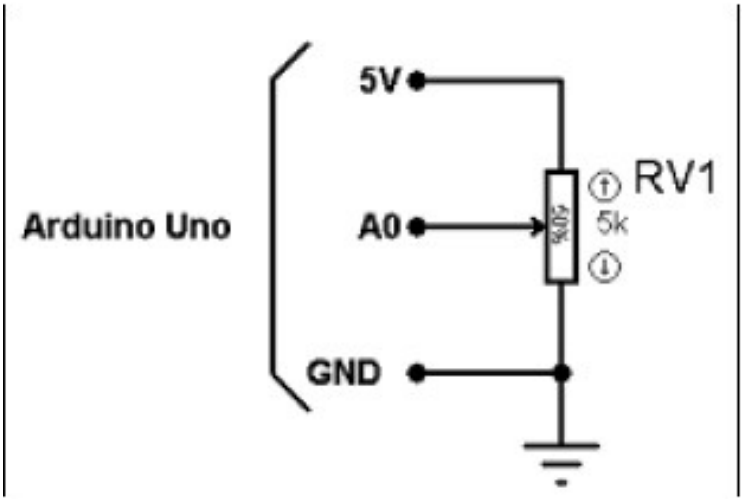

Figura 25. Diagrama de circuito.

- Se inició Visual Basic generando el siguiente formulario, e introduciendo el código fuente en el mismo:

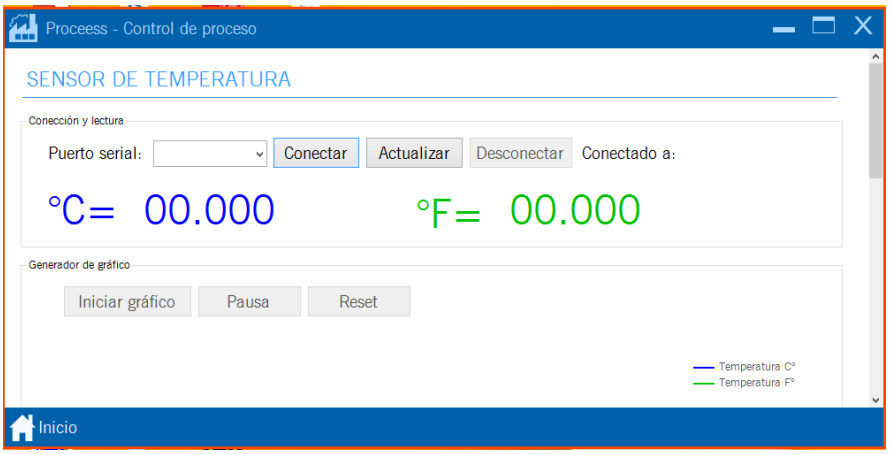

Figura 26. Formulario.

- Codificación de algoritmos

```
----------------------------------------------------------------------------------
Imports System.Runtime.InteropServices
Imports System.Windows.Forms.DataVisualization.Charting
<DllImport("user32.dll")>
  Public Shared Function ReleaseCapture() As Boolean
  End Function

  <DllImport("user32.dll")>
```

```
 Public Shared Function SendMessage(ByVal hWnd As IntPtr,
ByVal Msg As Integer, ByVal wParam As Integer, ByVal lParam
As Integer) As Integer
    End Function

    Private Const WM_NCLBUTTONDOWN As Integer = &HA1
    Private Const HTBORDER As Integer = 18
    Private Const HTBOTTOM As Integer = 15
    Private Const HTBOTTOMLEFT As Integer = 16
    Private Const HTBOTTOMRIGHT As Integer = 17
    Private Const HTCAPTION As Integer = 2
    Private Const HTLEFT As Integer = 10
    Private Const HTRIGHT As Integer = 11
    Private Const HTTOP As Integer = 12
    Private Const HTTOPLEFT As Integer = 13
    Private Const HTTOPRIGHT As Integer = 14

Private Sub Timer1_Tick(sender As Object, e As EventArgs)
Handles timerTemperature.Tick
        PuertoArduino.Write(Chr(10))
        lblTempC.Text = PuertoArduino.ReadExisting
        lblTempF.Text = (Val(lblTempC.Text) * 9 / 5 + 32)
        TEMPERATURAf = (Val(lblTempC.Text) * 9 / 5 + 32)
        TEMPERATURAc = (Val((lblTempF.Text) - 32) * 5 / 9)
    End Sub

If ListComPortSerial.SelectedItem = "" Then
            MsgBox("Seleccione un puerto de la lista.",
MsgBoxStyle.Information, Title:="Aviso")
        Else
            Try
                PuertoArduino.Close()
                PuertoArduino.PortName =
ListComPortSerial.SelectedItem
                PuertoArduino.Open()
                MsgBox("Conexion exitosa",
MsgBoxStyle.Information, Title:="Aviso")
                Label1.Text = "Conectado a: " +
ListComPortSerial.SelectedItem.ToString

                cmdDesconectar.Enabled = True
                ListComPortSerial.Enabled = False
                cmdConectar.Enabled = False
                timerTemperature.Enabled = True

                cmdIniciar.Enabled = True
                CmdPausar.Enabled = False
                cmdReset.Enabled = False
            Catch ex As Exception
```

```
                MsgBox("No se pudo establecer la conexion.",
MsgBoxStyle.Information, Title:="Aviso")

                cmdConectar.Enabled = True
                cmdDesconectar.Enabled = False
                ListComPortSerial.Enabled = True
                timerTemperature.Enabled = False
            End Try
        End If
End Class
```

- Conexión del cable USB y descargue el sketch al Arduino. Ejecución del programa en Visual Basic y lectura del valor que se presenta en la ventana.
- Se varía el sensor y se observe el cambio el valor visualizado en pantalla.

c. Conexión de sensor

Se realizó la conexión directamente el sensor con la placa Arduino mediante la interpuesta una resistencia, para su lectura correspondiente.

3.7.5. Etapa V.- Desarrollo del proceso de elaboración de néctar

Elaboración de néctar con la asistencia de software

Primero se realizó la programación, estructuración y desarrollo del software orientado a la elaboración de néctar, con el software ya instalado se procede a ejecutar dentro de la computadora y a seguir las instrucciones paso a paso, para el correcto uso del software se remite al manual de procedimiento de uso, donde específica a detalle los pasos a seguir.

Elaboración de néctar sin la asistencia de software

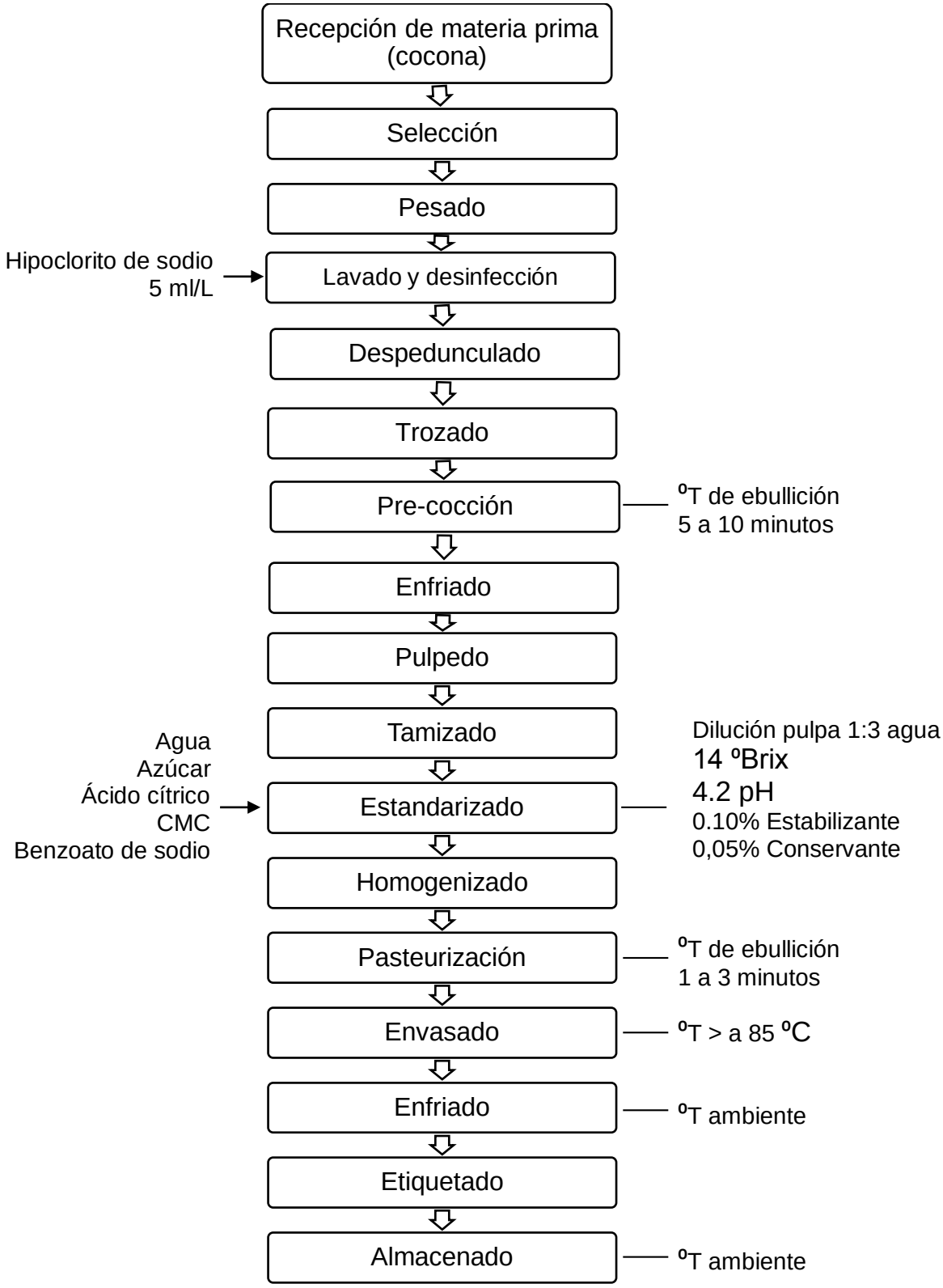

Figura 27. Diagrama de flujo, elaboración de néctar.

Descripción del proceso

a. Recepción de materia prima

En esta etapa se recepcionó la cocona en javas procedentes de la cosecha. Las frutas se reciben sin importar el tamaño de las mismas, siendo esto una ventaja de este tipo de productos al permitir usarlos de diferentes tamaños.

b. Selección

Se procedió a separar frutas malogradas, dañadas y defectuosas de los sanas. Esta operación es fundamental para obtener un producto de calidad. Ya que un producto de calidad necesita materia prima de calidad.

c. Pesado

Consistió en pesar las javas de cocona en una balanza digital. Esta operación se realizó para determinar el rendimiento.

d. Lavado y desinfección

En esta operación se procedió a sumergir la fruta en tinas con solución de agua más hipoclorito de sodio, con una concentración de 5 ml. /L. para desinfectar la fruta (cocona), después se enjuagó con abundante agua, de esta manera eliminar los residuos del desinfectante, que pueden causar alteraciones en operaciones posteriores.

Aleman (2015) señala que la desinfección se realiza con una concentración de 5 ml. /L. para desinfectar frutas.

e. Despedunculado

Después de eliminar el pedúnculo adherido al fruto, se cortó la parte superior. El corte se realiza mediante cualquier instrumento cortante inoxidable.

f. Trozado

Se realizó en forma manual empleando cuchillos, el trozado se realiza para facilitar el licuado.

g. Pre-cocción

Se realizó para uniformizar el color de la fruta a una temperatura de ebullición durante 5 – 10 minutos, para facilitar el pulpeado o licuado para ello se adiciona la dilución 1:3 de agua para realizar el escaldado.

Aleman (2015) señala precocer la fruta de 5 a 10 minutos.

h. Enfriado

Se enfrió a temperatura que aguante la manipulación; sin dañar ningún equipo ya sea la palpadura o licuadora.

i. Pulpeado

Consistió en reducir el tamaño la cocona ya escaldada hasta el mínimo tamaño. Esta operación se realiza con licuadora industrial o pulpeadora dependiendo del volumen del producto.

j. Tamizado

Esta operación consistió en reducir el tamaño de las partículas de la pulpa, otorgándole una apariencia más homogénea. Las pulpeadoras mecánicas o manuales facilitan esta operación por que cuentan con mallas de menor diámetro de abertura. En el caso de realizar el pulpeado con una licuadora, es necesario el uso de un tamiz para refinar la pulpa.

k. Estandarizado

En esta operación se realizó la mezcla de todos los ingredientes que constituyen el néctar.

Aleman (2015) menciona que se debe estandarizar la pulpa : agua con 1:3.

La estandarización involucra los siguientes pasos:

- **Dilución de la pulpa**

  Para calcular el agua a emplear se utilizó relaciones o proporciones representadas de la siguiente manera. Por ejemplo:

Dilución 1 : 3

Donde 1, significa “una” parte de pulpa o jugo puro de la fruta y 3, significa “tres” partes de agua, es decir estamos utilizando la relación “uno a tres”. La cantidad de agua varía de acuerdo a la fruta. Observemos las relaciones de dilución en el cuadro siguiente.

Tabla 6. *Dilución pulpa : agua*

FRUTA	**DILUCIÓN PULPA : AGUA**
Maracuyá	1 **:** 4 – 5
Granadilla	1 **:** 2 - 2.5
Cocona	1 **:** 3 – 5
Piña	1 **:** 2 – 2.5
Guanábana	1 **:** 3 – 3.5
Manzana	1 **:** 2 – 3
Durazno (blanquillo)	1 **:** 2 - 2.5
Uva Borgoña	1 **:** 2 – 3
Tamarindo	1 **:** 6 – 12
Poro	1 **:** 4.5
Mango	1 **:** 2.5 – 3
Berenjena	1 **:** 5
Tuna	1 **:** 3
Mora	1 **:** 3

Por ejemplo: Se tiene 5 kilos de pulpa de mango, la cual debe ser diluida con agua. *“Si la dilución recomendada es de 1:3, la cantidad de agua que debemos agregar es 15 kilos de agua”.*

- **Regulación del dulzor**

  Todas las frutas tienen su azúcar natural, sin embargo al realizar la dilución con el agua ésta tiende a bajar. Por esta razón es necesario agregar azúcar hasta un rango que puede variar entre los 13 a 18 °Brix.

Los grados Brix 14 representan el porcentaje de sólidos solubles presentes en una solución.

- **Regulación de la acidez**

  El ácido cítrico al igual que el azúcar es un componente de las frutas, sin embargo esta también disminuye al realizarse la dilución. En tal sentido es necesario que el producto tenga un pH adecuado que contribuya a la duración del producto.

- **Adición del estabilizado**

  En el siguiente cuadro se indica la cantidad de estabilizante que se requiere para los néctares de algunas frutas:

Tabla 7. *Adición de CMC*

Frutas	% de estabilizante CMC
Frutas pulposas Por ejemplo manzana, mango, Durazno	0,07%
Frutas menos pulposas Por ejemplo poro, granadilla, maracuyá	0,10 – 0,15%

Por ejemplo: si se aplica 0,10% de estabilizante CMC, significa que por cada kilo de dilución o néctar se aplicara 1 gramo de estabilizante CMC. “Entonces para 10 kilos de néctar de granadilla se añadirán 10 gramos de CMC”.

Para facilitar la disolución del CMC en el néctar, se debe mezclar previamente con el azúcar, y agregar al néctar momentos antes que llegue al punto de ebullición, para así evitar la formación de grumos.

- **Adición del conservante**

  La cantidad de agente conservante a adicionar no debe ser mayor al 0.05% del peso del néctar.

Resulta muy importante tener en cuenta la siguiente recomendación al momento realizar la operación de estandarización:

"Los cálculos que se realizan para la formulación del néctar, deben hacerse en función al peso de cada uno de los ingredientes. En tal sentido el cálculo de pulpa de fruta y agua se deben expresar en kilogramos o sus equivalencias".

l. Homogenizado

Esta operación tuvo por finalidad uniformizar la mezcla. En este caso consiste en remover la mezcla hasta lograr la completa disolución de todos los ingredientes.

m. Pasteurización

Esta operación se realizó con la finalidad de reducir la carga microbiana y asegurar la inocuidad del producto.

Calentamiento del néctar hasta su punto de ebullición, manteniéndolo a esta temperatura por un espacio de 1 a 3 minutos.

Luego de esta operación se retiró del fuego, se separó la espuma que se forma en la superficie y se procede inmediatamente al envasado.

Aleman (2015) señala que se debe pasteurizar hasta su punto de ebullición por un espacio de 1 a 3 minutos.

n. Envasado

El envasado se realizó en caliente, a una temperatura no menor a 85°C. El llenado del néctar es hasta el tope del contenido de la botella, evitando la formación de espuma. Inmediatamente se coloca la tapa, la cual se realiza de forma manual en el caso que se emplee las tapas denominadas "taparosca".

Si durante el proceso de envasado la temperatura del néctar disminuye por debajo de 85°C, se debe detener esta operación. Se procede a calentar el néctar hasta su temperatura de ebullición, para proseguir luego con el envasado.

Al enfriarse el producto, ocurrirá la contracción del néctar dentro de la botella, lo que viene a ser la formación de vacío, esto último representa el factor más importante para la conservación del producto.

Aleman (2015) señala que el envasado debe realzarse a una temperatura no menor a 85°C.

o. Enfriado

Se efectuó inmediatamente después del envasado, sumergiendo los envases, en contenedores de agua fría con circulación continua. La temperatura del agua potable circulante es menor del ambiente estando aproximadamente en 17 °C.

El enfriado se realiza con chorros de agua fría, que a la vez nos va a permitir realizar la limpieza exterior de las botellas de algunos residuos de néctar que se hubieran impregnado.

p. Etiquetado

El etiquetado constituye la etapa final del proceso de elaboración de néctares. En la etiqueta se incluyó toda la información sobre el producto.

q. Almacenado

Los néctares fueron almacenados a temperatura ambiente, en un lugar fresco y colocados sobre parihuelas.

Aleman (2015) señala que el almacenado debe realizarse a una temperatura ambiente.

3.7.6. Etapa VI.- Determinación de las características fisicoquímicas y sensoriales del néctar y análisis estadístico

a. Evaluación de aceptación del producto software

Se utilizó esta prueba para observar la aceptación del producto, para ajustarse a un modelo de chi-cuadrado se utilizó como el más adecuado la prueba de bondad de ajuste.

La población para la encuesta estaba conformado por los estudiantes de la universidad nacional Hermilio Valdizán

b. Evaluación de la influencia de la programación en la elaboración de productos

Se utilizó esta prueba de asociación u homogeneidad para determinar si el valor observado de una variable depende del valor observado de otra variable, o para determinar si una variable está asociada a otra variable. En este caso se compara la programación en el desarrollo de software utilizando calidad de información con la elaboración del producto con dicha información. Comparar si existe relación o asociación entre ambos es decir si al utilizar la información de calidad estructurado en el software conlleva a obtener productos de buena calidad.

c. Evaluación de comparación entre el néctar elaborado con la utilización del software con néctar elaborado sin la utilización de software

Se realizan tres pruebas independientes de evaluación sensorial respecto al color, olor y sabor.

Prueba t para muestras independientes compara las medias de dos grupos de casos. En ese caso comparar el néctar elaborado con la utilización del software con néctar elaborado sin la utilización de software.

IV. RESULTADOS

4.1. Etapa I.- Análisis de requisitos

Esta primera etapa estuvo enfocada a analizar, comprender, conocer y entender el funcionamiento del proceso de elaboración de néctar en su totalidad, y asimismo establecer los requisitos que cubrirá el software. Para esto, se emplearon las técnicas de recolección de datos, como las entrevistas no estructuradas, la observación directa y la revisión documental.

Con la información obtenida se elaboró los siguientes documentos:

a) Plan de iteración
b) Documento visión
c) Especificación de casos de uso

La elaboración de los documento fue de acuerdo a la metodología RUP, usando los artefactos necesarios y básicos para la presente investigación.

a. Plan de iteración

A continuación se muestra el plan de iteración de manera básica, siguiendo la metodología RUP.

Proyecto: “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Nombre del documento:	Versión:	Fecha:
Plan de Iteración	1.0	Julio 2016

Proyecto:
“Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Proyecto: “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Nombre del documento:	Versión:	Fecha:
Plan de Iteración	1.0	Julio 2016

Plan de Iteración

Introducción

Propósito

El propósito es presentar el plan de general del proyecto de “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”.

Alcance

Este plan de Iteración se aplica al proyecto “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”.

Definiciones, acrónimos y abreviaturas

Documento Glosario

Referencias

Documento visión.

Documento plan de riesgos.

Documento glosario.

Resumen

La Iteración Preliminar desarrollará los requisitos del producto y establecerá el caso del proyecto “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”. Se desarrollarán los principales casos de uso y el Plan de proyecto de alto nivel.

Proyecto: “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Nombre del documento:	Versión:	Fecha:
Plan de Iteración	1.0	Julio 2016

Plan

Tareas de iteración

Tabla 8. *Tareas de iteración*

	tarea	Duración	Comienzo	Fin
1	**PLAN DE ITERACIÓN GENERAL**	**537 Días**	**01/07/2016**	**20/12/2017**
2	**ANÁLISIS DE REQUISITOS**	**133 Días**	**01/07/2016**	**11/11/2016**
3	evaluar el estado del proceso	10 Días	01/07/2016	11/07/2016
4	identificación legal del proceso	3 Días	12/07/2016	15/07/2016
5	identificación del proceso de néctar	0 Días	16/07/2016	16/07/2016
6	identificación tecnológica del proceso	0 Días	16/07/2016	16/07/2016
7	identificación digital del proceso	0 Días	16/07/2016	16/07/2016
8	identificación real del proceso	0 Días	16/07/2016	16/07/2016
9	evaluación del alcance del proyecto	0 Días	16/07/2016	16/07/2016
10	evaluación de riesgos del proyecto	0 Días	16/07/2016	16/07/2016
11	**Requerimientos**	**58 Días**	**17/07/2016**	**13/09/2016**
12	analizar del problema	0 Días	17/07/2016	17/07/2016
13	Análisis del proceso de néctar	0 Días	17/07/2016	17/07/2016
14	definir el software a desarrollar	11 Días	17/07/2016	28/07/2016
15	definir los requisitos de software	22 Días	29/07/2016	20/08/2016
16	entender los requerimientos de los usuarios	0 Días	20/08/2016	20/08/2016
17	documento visión	9 Días	21/08/2016	30/08/2016
18	documento glosario	12 Días	01/09/2016	13/09/2016
19	**Análisis y diseño**	**43 Días**	**14/09/2016**	**27/10/2016**
20	modelo de caso de usos	1 Días	14/09/2016	15/09/2016
21	especificaciones de caso de uso	5 Días	16/09/2016	21/09/2016
22	arquitectura de diseño	35 Días	22/09/2016	27/10/2016

23	**gestión del proyecto**	**14 Días**	**28/10/2016**	**11/11/2016**
24	plan de iteración	2 Días	28/10/2016	30/10/2016
25	plan d administración de riesgos	12 Días	30/10/2016	11/11/2016
26	**DISEÑO DE LA ARQUITECTURA**	**31 Días**	**11/11/2016**	**12/12/2016**
27	definición del diseño	3 Días	11/11/2016	14/11/2016
28	definición del producto	5 Días	14/11/2016	19/11/2016
29	definición d la arquitectura	4 Días	19/11/2016	23/11/2016
30	Modelo de Casos de Uso del Sistema	7 Días	23/11/2016	30/11/2016
31	Especificación de casos de uso del sistema	12 Días	30/11/2016	12/12/2016
32	**CONSTRUCCIÓN DEL SOFTWARE**	**373 Días**	**12/12/2016**	**20/12/2017**
33	Codificación	185 Días	12/12/2016	15/06/2017
34	integración	60 Días	16/06/2017	15/08/2017
35	pruebas unitarias	0 Días	15/08/2017	15/08/2017
36	pruebas de integración	0 Días	15/08/2017	15/08/2017
37	modelo de despliegue	3 Días	16/08/2017	19/08/2017
38	aplicación de prueba final	0 Días	19/08/2017	19/08/2017
39	instalación	0 Días	19/08/2017	19/08/2017
40	corrección de errores	10 Días	19/08/2017	29/08/2017
41	operación y mantenimiento	113 Días	29/08/2017	20/12/2017

Recursos

Se requiere la revisión de las normas que establecen el proceso de elaboración de néctar, como también reuniones con alumnos, docentes y otros profesionales.

Recursos humanos

Para la presente investigación se cuenta con la dirección y asesoramiento del Mg. Gregorio Cisneros Santos, asimismo para la presente investigación hemos realizado diversos cursos extras de programación para la obtención de mejores resultados.

Recursos financieros

El presupuesto para el proyecto es de **S/. 7706.00**, solo en materiales y equipos.

Criterios de evaluación

Se evaluara la eficiencia del software en la aplicación de elaboración de néctar.

b. Documento visión

A continuación se muestra el documento visión de manera básica, siguiendo la metodología RUP.

Proyecto: “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Nombre del documento:	Versión:	Fecha:
Documento visión	1.0	Agosto 2016

Proyecto:
“Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Documento visión.
Versión 1.0

Proyecto: “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Nombre del documento:	Versión:	Fecha:
Documento visión	1.0	Agosto 2016

Visión

- **Introducción**

a. Objetivo

El objetivo de este documento es establecer la definición inicial para desarrollar un software para automatizar y sistematizar los datos en la elaboración de néctar. Para lograr este propósito es necesario describir brevemente el entorno del proyecto, definiendo las necesidades, las características y estableciendo los requerimientos que sirvan de base para la planificación de este proyecto.

b. Alcance

El software se desarrollara en el lenguaje de Visual Basic .NET, creando un producto software para el control eficiente en el proceso de elaboración de néctar.

c. Definiciones, acrónimos y abreviaturas

RUP: Proceso Unificado Rational.

UML: Lenguaje Unificado de Modelado.

Proyecto: Conjunto de acciones que conducen a un fin determinado.

Sistema automatizado: sistema donde se transfieren tareas de producción, realizadas habitualmente por operadores humanos a un conjunto de elementos tecnológicos.

d. Referencias

Plan de iteración.

Documento glosario.

Proyecto: “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Nombre del documento:	Versión:	Fecha:
Documento visión	1.0	Agosto 2016

- **Posicionamiento**

a. Oportunidad de negocio

Las oportunidades que se presentan con el desarrollo de este proyecto son:

- Manejo adecuado de la información oportunamente.
- Sistematización y automatización de la información.
- Reporte de datos precisos.
- Formulación correcta.
- Aumento de la eficiencia.
- Actualización a herramientas tecnológicas.

b. Declaración de problema

Tabla 9. *Planteamiento del problema*

Problema	**Conceptualización**
El problema es	La realización manual de operaciones matemáticas, y la falta de información de alta calidad bien estructura para llevar todo el proceso de elaboración de néctar.
Afecta a	Las micro empresas, planas agroindustriales, instituciones.
Cuyo impacto es	La generación de resultados infavorables, como la obtención de productos de mala calidad.
Una adecuada solución sería	El desarrollo de un software para la automatización de la información d ala calidad y disponer oportunamente.

Proyecto: “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Nombre del documento:	Versión:	Fecha:
Documento visión	1.0	Agosto 2016

c. Declaración de posición de producto

Tabla 10. *Posición de producto*

Para	Empresas Microempresas Instituciones
Quienes	Requieren datos confiables y de calidad.
El software	Es una aplicación de escritorio, desarrollada en Visual Basic .NET
Funcionalidad	Permitirá realizar procesamiento de datos en un tiempo muy breve, arrojando respuestas seguras, y permitirá realizar el control del proceso de elaboración de néctar a través de los aplicativos incorporados.
No como	Actualmente se realiza de manera manual, con un riesgo en los datos para el proceso.
Nuestro producto	Influirá n la eficiencia en el proceso de elaboración de néctar.

- **Descripción de participantes y usuarios**

Proyecto: “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Nombre del documento:	Versión:	Fecha:
Documento visión	1.0	Agosto 2016

Roles y responsabilidades de los participantes

Un rol consiste en las obligaciones que implican una determinada función o posición que tiene un individuo dentro de un proyecto. Un rol es el papel que ejerce un actor en una actividad o proyecto, o de un grupo de actores trabajando juntos como un equipo. Un actor puede desempeñar diversos roles, así como un mismo rol puede ser representado por varios actores.

Las responsabilidades de un rol son ejecutar un conjunto de actividades y el ser el dueño de un conjunto de artefactos.

Antes de asignar los roles de cada participante, se describirán cada uno de los roles que se utilizarán en el proyecto:

Tabla 11. *Roles y responsabilidades*

Rol	Responsabilidad
Asesor del proyecto	Dirigir el proceso de investigación.
Analista de sistemas	Captura, especificación y validación de requisitos, encargado del desarrollo de aplicaciones en lo que respecta a su diseño y obtención de los algoritmos, así como de analizar las posibles utilidades y modificaciones necesarias de los sistemas operativos para una mayor eficacia de un sistema informático. Otra misión de estas personas es dar apoyo técnico a los usuarios.
Integrador	Es el responsable de planear y realizar la integración de elementos de la Aplicación.

Programador	Se dedica a uno o más aspectos del proceso de desarrollo de software. Se trata de un ámbito más amplio de la programación algorítmica.
Especialista en pruebas (tester)	Es responsable de realizar las pruebas del software cada vez que se realice una iteración y las pruebas finales anotando los resultados de esa comprobación.

Proyecto: “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Nombre del documento:	Versión:	Fecha:
Documento visión	1.0	Agosto 2016

Tabla 12. *Participantes y roles*

Mg. Gregorio Cisneros Santos	Asesor de la investigación
Bach. Franklin Luis Salazar Cano	Ejecutor de la investigación. Analista Programador Integrador Mantenimiento

- **Necesidades clave de participantes o usuarios**

  Necesidades de participantes a nivel de Sistemas

Tabla 13. *Necesidades de participantes a nivel de sistemas*

Necesidad	**Prioridad**	**Soluciones propuestas**
Captura y toma de requerimientos.	Alta	Recolección de datos e información.
Desarrollo del software	Alta	Integración y codificación del software
Diseño de la arquitectura del software	Alta	Diseñar la arquitectura mediante el IDE de Visual Studio.

Proyecto: “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Nombre del documento:	Versión:	Fecha:
Documento visión	1.0	Agosto 2016

a. Necesidades de los usuarios

Tabla 14. *Necesidades de los usuarios*

Necesidad	Prioridad	Soluciones propuestas
Disponer de información confiable y segura	Alta	Estructurar la información en el software
Dosificar la formulación eficientemente	Alta	Desarrollar aplicaciones con algoritmos matemáticos para la formulación
Controla el proceso eficientemente	Alta	Desarrollar aplicaciones con una arquitectura dinámica.

b. Descripción global del producto

Perspectiva del producto

El producto o software a desarrollar es para realizar un proceso eficiente en la elaboración de néctar.

Resumen de capacidades

A continuación se mostrará un listado de las capacidades que ofrecerá el producto:

Proyecto: “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Nombre del documento:	Versión:	Fecha:
Documento visión	1.0	Agosto 2016

Tabla 15. *Capacidades del software*

Beneficios	**Funcionalidades**
Formulación eficiente	Se podrá realizar la dosificación de los insumos con datos confiables.
Control del proceso	Se realizar el control del proceso a través de los aplicativos del software
Sistematización de datos	El software brindara datos estructurados que permitirán realizar el proceso de elaboración de néctar eficientemente.

Presunciones y dependencias

Se asume que el alcance expresado en primer término para este documento no va a ser ampliado. En caso de que ocurriera una ampliación del alcance habría que realizar modificaciones en las funcionalidades.

Licenciamiento e instalación

El software está desarrollado para el sistema operativo de Windows en todas sus versiones.

Características del producto

El software presenta las siguientes características:

Corrección, usabilidad / facilidad de aprendizaje, integridad, fiabilidad, eficiencia, seguridad.

Proyecto: “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Nombre del documento:	Versión:	Fecha:
Documento visión	1.0	Agosto 2016

- **Requerimientos mínimos del proyecto**

Requerimientos de software

Tabla 16. *Requerimientos de software*

Software	**Requerimiento**
Sistema operativo	Windows XP, 7, 8 y 10

Requerimientos de hardware

Tabla 17. *Requerimientos de hardware*

Hardware	**Requerimiento**
Memoria RAM	Min. 500 MB
Disco duro	Min. 10 GB
Resolución de pantalla	Min. 1024 x 768 px

- **Restricciones**

  El software es libre no posee licencia ni claves para su operación.

c. Especificación de casos de uso

A continuación se muestra el documento de especificaciones de caso de uso, siguiendo la metodología RUP.

Proyecto: “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Nombre del documento:	Versión:	Fecha:
Especificaciones de Casos de Uso	1.0	Agosto 2016

Proyecto:
“Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Especificaciones de Casos de Uso.
Versión 1.0

Proyecto: “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Nombre del documento:	Versión:	Fecha:
Especificaciones de Casos de uso	1.0	Agosto 2016

- **Introducción**

a. Propósito

El software realiza la función de recolectar datos, procesar, y mostrar resultados, para la realización de formulación de insumos, control del proceso, sistemazación de la información.

b. Alcance

Proceso de elaboración de néctar.

c. Definiciones, Acrónimos y Abreviaturas

Documento glosario.

d. Referencia

Documento glosario.

Documento visión.

e. Resumen

Este documento muestra de manera básica y general el funcionamiento del software en la elaboración de néctar.

- **Flujo de eventos**

Para el funcionamiento adecuado del software seguir los siguientes pasos:

Figura 28. Flujo de operación del software.

4.2. Etapa II.- Diseño de la arquitectura

Se logró diseñar una adecuada arquitectura que nos permite realizar el proceso de manera dinámica con una interfaz gráfica atractiva.

Figura 29. Interfaz gráfica del software.

El diseño de la arquitectura del software permite operar en las funcionalidades guiando al usuario correctamente.

4.3. Etapa III.- Diseño y desarrollo del software para elaboración de néctar

Se realizó exitosamente el desarrollo del producto software para la elaboración de néctar. El software se construyó de acuerdo a las necesidades del usuario.

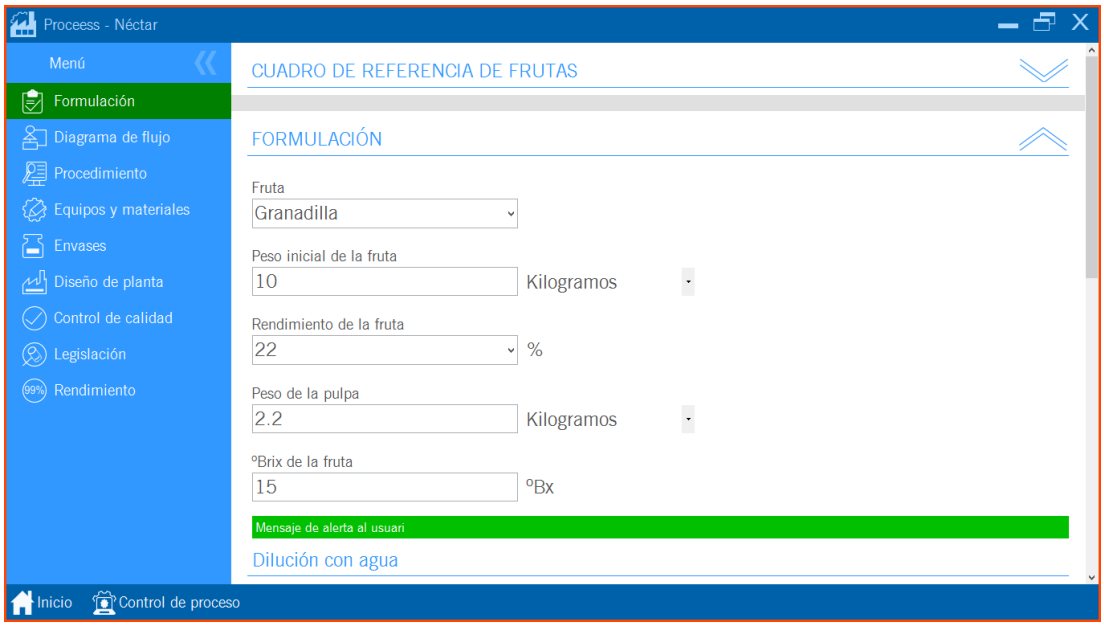

Figura 30. Ejecución del software.

Se acompaña con un documento glosario, que se muestra en anexos.

4.4. Etapa IV.- Diseño y desarrollo del software para control del proceso

A continuación se muestra el grafico de la integración de los sensores directamente con el código fuente del software, mediante los algoritmos propios del software, declarados en su estructura.

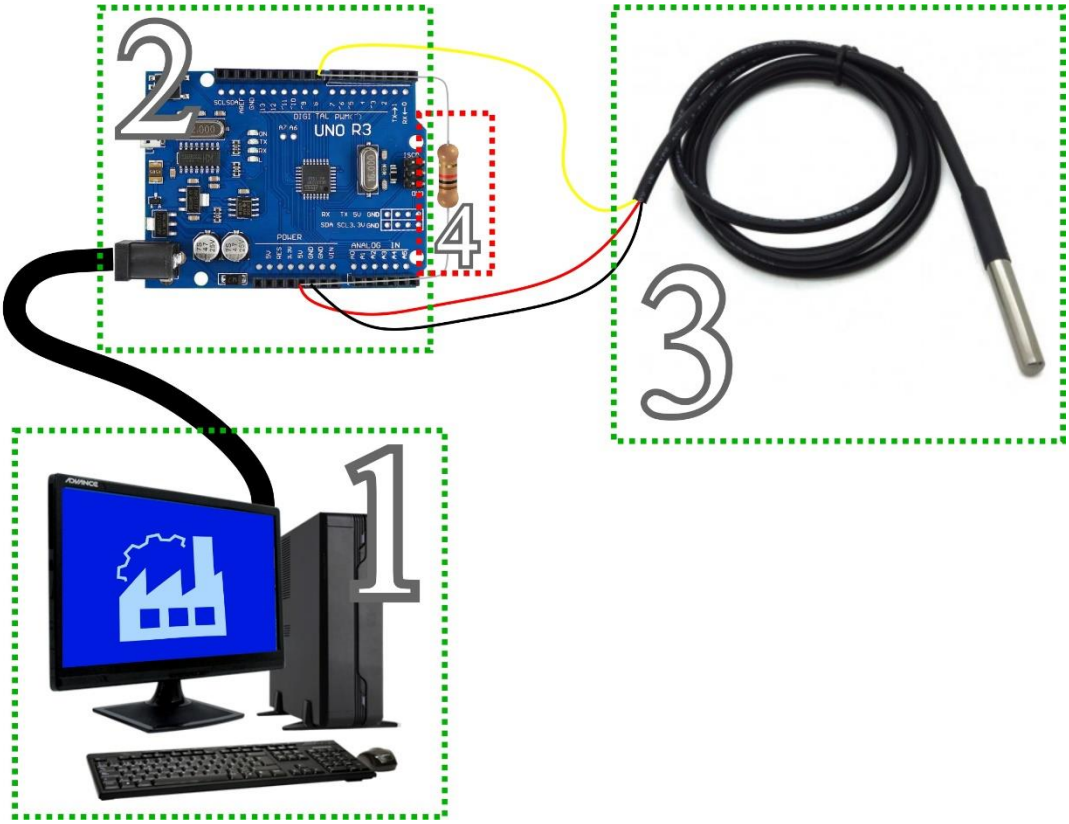

Figura 31. Integración del software con los sensores.

Se conectó exitosamente los dispositivos periféricos con la computadora, mediante la codificación de los algoritmos, lo que hicieron posible la lectura y reconocimiento de datos de los sensores.

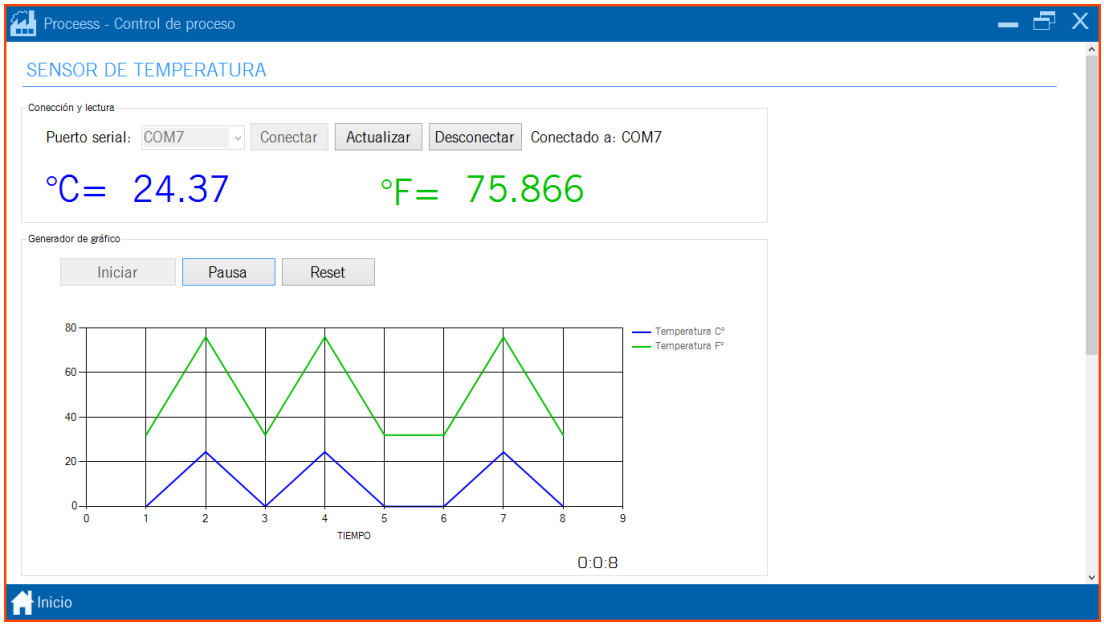

Figura 32. Formulario del software.

Se logró integrar satisfactoriamente el sensor de temperatura mediante la placa Arduino.

4.5. Etapa V.- Desarrollo del proceso de elaboración de néctar

a. Elaboración de néctar con la asistencia del software

Se elaboró exitosamente el néctar con la asistencia del software.

Figura 33. Elaboración de software bajo la asistencia del software desarrollado.

b. Elaboración de néctar sin la asistencia del software

Se realizó la elaboración de néctar sin la asistencia del software, realizando las operaciones de cálculos de manera manual, teniendo como referencia la Norma General para Zumos (jugos) y Néctares de Frutas (codex stan 247-2005), también con la consideración del Código Internacional Recomendado de Prácticas - Principios Generales de Higiene de los Alimentos (CAC/RCP 1-1969), Norma para los Azúcares (CX-STAN 212-1999), Directrices de la OMS para la Calidad del Agua Potable (Volúmenes 1 y 2), Norma General para los Aditivos Alimentarios (NGAA), Principios para el Establecimiento y la Aplicación de Criterios Microbiológicos para los Alimentos (CAC/GL 21-1997), Norma General para el Etiquetado de los Alimentos Preenvasados (CODEX STAN 1-1985), Y las Directrices para el Uso de Declaraciones de Propiedades Nutricionales (CAC/GL 23-1997) y las Directrices sobre Etiquetado Nutricional (CAC/GL 2-1985).

Figura 34. Elaboración de software sin la asistencia del software.

4.6. Etapa VI.- Determinación de las características fisicoquímicas y sensoriales del néctar y análisis estadístico

Se realizó el análisis estadístico con los siguientes:

a. Determinación de las características fisicoquímicas del néctar
b. Prueba Chi-cuadrado: Aceptación del producto
c. Prueba Chi-cuadrado de asociación u homogeneidad: influencia de la programación en la elaboración de productos.
d. Prueba T muestras independientes: Comparar el néctar elaborado con la utilización del software con néctar elaborado sin la utilización de software.

Se detallan a continuación cada prueba realizada.

4.6.1. Determinación de las características fisicoquímicas del néctar

Se realizó el análisis fisicoquímico del néctar elaborado bajo la asistencia del software, en el laboratorio de procesos alimentarios de la escuela académico profesional de ingeniería agroindustrial de la universidad Hermilio Valdizán, donde se obtuvo los siguientes resultados:

pH: 4.2
°Brix: 14.0

Los resultados obtenidos se encuentran dentro del rango aceptable por el CODEX STAN 247-2005.

4.6.2. Prueba Chi-cuadrado: aceptación del producto

Se utilizó la prueba de Chi cuadrado con bondad de ajuste para evaluar la aceptación del software por parte de los usuarios, para la investigación se utilizó 50 panelistas, los resultados muestran:

a. Formulación de las hipótesis nula y alternativa

H_0: el software desarrollado **no** tendrá aprobación de los usuarios.

H_0: $\mu_1 = \mu_2$

H_1: el software desarrollado **si** tendrá aprobación de los usuarios.

H_1: $\mu_1 \neq \mu_2$

b. Identificación

Cola derecha

c. Nivel de significancia, en condiciones normales se trabaja con un

$\alpha = 0.05$

d. Estadístico de la prueba, usaremos la prueba no paramétrica CHI-CUADRADO.

e. Esquema de la prueba

gl = 6

α= 5%

$x^2 = 12.59$

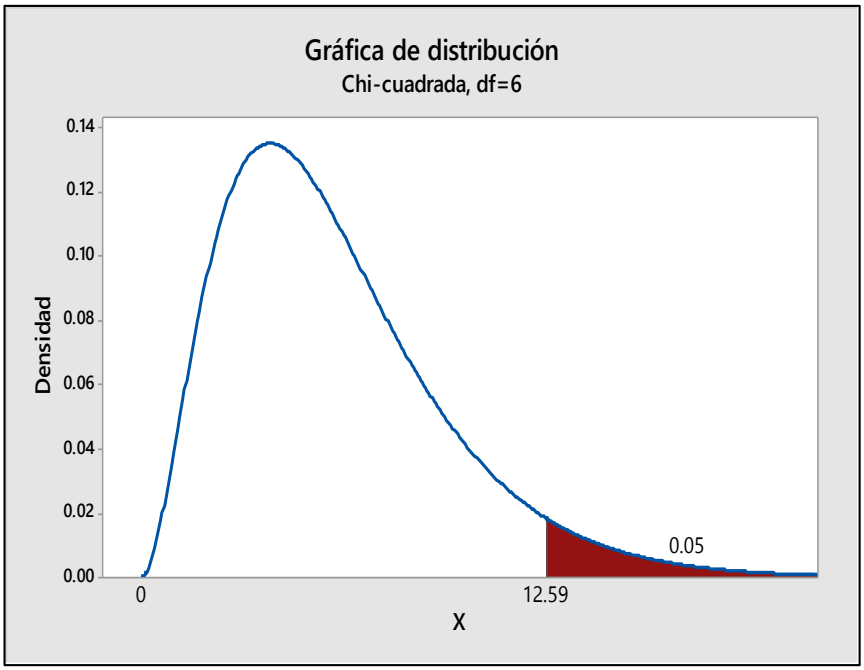

Figura 35. Distribución estadística chi-cuadrado, homogeneidad.

f. Cálculo del estadístico de la prueba

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

$$con\ [k-1]\ Grados\ de\ libertad$$

Tabla 18. *Cantidad de aceptación y rechazo*

Preguntas	ACEPTACIÓN	RECHAZO
1	49	1
2	50	0
3	48	2
4	49	1
5	50	0
6	46	4
7	40	10

Tabla 19. *Frecuencias esperadas*

Preguntas	Aceptación	Promedio	Rechazo	Promedio
1	49	47.4285714	1	2.5714286
2	50	47.4285714	0	2.5714286
3	48	47.4285714	2	2.5714286
4	49	47.4285714	1	2.5714286
5	50	47.4285714	0	2.5714286
6	46	47.4285714	4	2.5714286
7	40	47.4285714	10	2.5714286

Tabla 20. *Estadístico de contraste*

Preguntas	Aceptación	Promedio	Estadístico de contraste	Rechazo	Promedio	Estadístico de contraste
1	49	47.4285714	0.052065404	1	2.57142857	0.96031746
2	50	47.4285714	0.139414802	0	2.57142857	2.571428571
3	48	47.4285714	0.006884682	2	2.57142857	0.126984127
4	49	47.4285714	0.052065404	1	2.57142857	0.96031746
5	50	47.4285714	0.139414802	0	2.57142857	2.571428571
6	46	47.4285714	0.04302926	4	2.57142857	0.793650794
7	40	47.4285714	1.163511188	10	2.57142857	21.46031746
			1.596385542			**29.44444444**
				X2 = **31.04082999**		

Estadístico de contraste: $\chi^2 = \mathbf{31.04082999}$

g. Decisión

Como el estadístico de la prueba es mayor que el esquema de la prueba se acepta la hipótesis alternativa y rechazamos la hipótesis nula, por lo que el software desarrollado **si tiene aprobación de los usuarios.**

4.6.3. Prueba Chi-cuadrado de asociación u homogeneidad: influencia de la programación en la elaboración de productos

Se utilizó esta prueba de asociación u homogeneidad para determinar si el valor observado de una variable depende del valor observado de otra variable, o para determinar si una variable está asociada a otra variable. En este caso se compara la programación en el desarrollo de software utilizando calidad de información con la elaboración del producto con dicha información. Comparar si existe relación o asociación entre ambos es decir si al utilizar la información de calidad estructurado en el software conlleva a obtener productos de buena calidad.

a. Formulación de las hipótesis nula y alternativa

H0: el desarrollo del software con calidad de información no influirá en el proceso de elaboración del néctar.

H_0: $\mu_1 = \mu_2$

H1: el desarrollo del software con calidad de información si influirá en el proceso de elaboración del néctar.

H_1: $\mu_1 \neq \mu_2$

b. Identificación

Cola derecha

c. Nivel de significancia

$\alpha = 0.05$

d. Estadístico de la prueba, usaremos la prueba no paramétrica Chi-cuadrado de homogeneidad u asociación.

e. Se realiza el esquema de la prueba

gl = 1

α= 5%

$x^2 = 3.84$

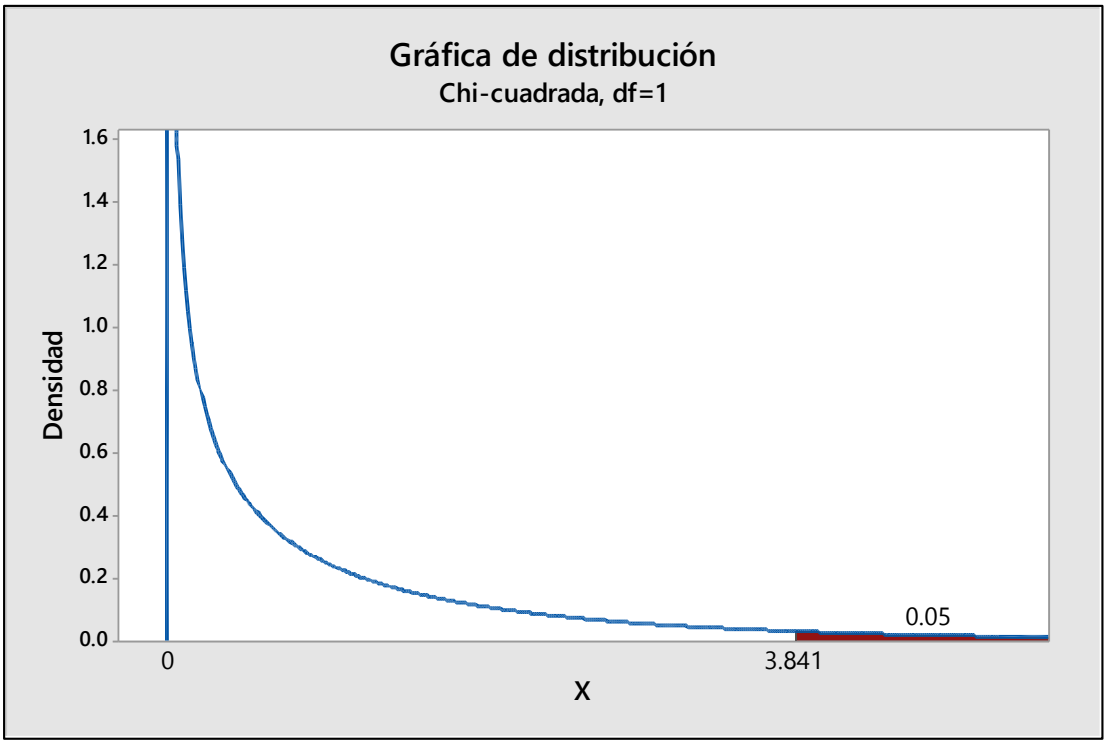

Figura 36. Distribución estadística chi-cuadrado.

f. Cálculo del estadístico de la prueba

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

$$con\ [n-1][k-1]\ Grados\ de\ libertad$$

Tabla 21. *Resumen del total de encuestados*

Productos	Aceptación	Rechazo	Total
Productos A - Con asistencia del software	38	12	50
Productos B - Sin asistencia del software	23	27	50

Tabla 22. *Frecuencias esperadas*

Productos	Aceptación	Promedio	Rechazo	Promedio
Productos A - Con asistencia del software	38	30.5	12	19.5
Productos B - Sin asistencia del software	23	30.5	27	19.5

Tabla 23. *Estadístico de contraste*

Productos	Aceptación	Promedio	Estadístico de contraste	Rechazo	Promedio	Estadístico de contraste
Productos A - Con asistencia del software	38	30.5	1.844262295	12	19.5	2.884615385
Productos B - Sin asistencia del software	23	30.5	1.844262295	27	19.5	2.884615385
			3.68852459			5.769230769
				X^2 = 9.457755359		

Estadístico de contraste: $\chi^2 = \mathbf{9.46}$

g. Decisión estadística

Como el estadístico de la prueba es mayor que el esquema de la prueba se acepta la hipótesis alternativa y rechazamos la hipótesis nula, por lo que el desarrollo del software con calidad de información **si influirá** en el proceso de elaboración del néctar.

4.6.4. Prueba T muestras independientes: Comparar el néctar elaborado con la utilización del software con néctar elaborado sin la utilización de software.

Se realizaron tres pruebas independientes de evaluación sensorial respecto al color, olor y sabor, se evaluó la diferencia significativa que existe entre ambos, es decir entre el néctar elaborado con la asistencia del software usando información estructurada de calidad el néctar elaborado convencionalmente.

Tabla 24. *Escala de evaluación cualitativa y cuantitativa*

Evaluación	**Producto**				
Evaluación cualitativa	EXCELENTE	MUY BUENO	BUENO	REGULAR	DEFICIENTE
Evaluación cuantitativa	5	4	3	2	1

4.6.4.1. Evaluación sensorial en color:

a. Hipótesis:

H_0: no existe diferencia significativa entre el color del néctar elaborado con la asistencia del software utilizando información de calidad y la elaboración del néctar sin la utilización de software realizado de manera convencional.

$H_0: \mu_1 = \mu_2$

H_1: existe diferencia significativa entre el color del néctar elaborado con la asistencia del software utilizando información de calidad y la elaboración del néctar sin la utilización de software realizado de manera convencional.

$H_1: \mu_1 \neq \mu_2$

b. **Nivel de significancia,** en condiciones normales se trabaja con un α = 0.05.

c. **Estadístico de la prueba,** prueba t para muestras independientes, utilizando la herramienta de IBM SPSS Statistics 22.

d. **Prueba de Levene** prueba de igualdad de varianza.

Hipótesis

H_0: se asumen varianzas iguales

H_0: $\mu_1 = \mu_2$

H_1: no se asumen varianzas iguales

H_1: $\mu_1 \neq \mu_2$

Igualdad de varianza:

Sig. = 0.000

α= 5% = 0.05

Tabla 25. *Prueba de Levene, evaluación de color*

		Prueba de Levene de calidad de varianzas	
		F	Sig
PUNTUACIONES	Se sumen varianzas iguales	41,013	0,000
	No se asumen varianzas iguales		

Conclusión

Como la significancia es menor que el alfa, aceptamos la hipótesis alternativa o de trabajo y rechazamos la hipótesis nula, por lo que **no se asumen varianzas iguales.**

e. **Cálculo del estadístico de la prueba**

En la siguiente tabla se muestra la media y la desviación estándar, donde se aprecia que hay una gran diferencia en los promedios de los productos,

donde el **producto a** se muestra por encima del **producto b,** por lo que se asume que el producto a obtuvo mejores calificaciones.

Tabla 26. *Estadístico de grupo, evaluación de color*

estadísticas de grupo					
	PRODUCTOS	N	Media	Desviación estándar	Media de error estándar
PUNTUACIONES	PRODUCTO A	50	83,500	19,30555	2,73022
	PRODUCTO B	50	69,000	35,19624	4,97750

En la siguiente tabla se muestra la significancia bilateral, el valor a utilizar es del segundo dato, porque no se asumen varianzas iguales.

Tabla 27. *Prueba de muestras independientes, prueba t para igualdad de medias*

prueba t para igualdad de medias						
					95% de intervalo de confianza	
t	gl	Sig. (bilateral)	Diferencia de medias	Diferencia de error estándar	Inferior	Superior
2,554	98	,12	14,50000	5,67711	3,23396	25,76604
2,554	76,037	,13	14,50000	5,67711	3,19314	25,80686

f. Decisión estadística

Significancia bilateral: 0.013

Valor de alfa: 0.05

Tenemos el valor de la significancia que es menor que el valor del alfa, por lo tanto aceptamos la hipótesis alternativa por lo que se concluye que existe diferencia significativa entre el color del néctar elaborado con la

asistencia del software utilizando información de calidad y la elaboración del néctar sin la utilización de software realizado de manera convencional.

4.6.4.2. Evaluación de olor

a. Formulación de las hipótesis nula y alternativa

H_0: no existe diferencia significativa entre el olor del néctar elaborado con la asistencia del software utilizando información de calidad y la elaboración del néctar sin la utilización de software realizado de manera convencional.

$H_0: \mu_1 = \mu_2$

H_1: existe diferencia significativa entre el olor del néctar elaborado con la asistencia del software utilizando información de calidad y la elaboración del néctar sin la utilización de software realizado de manera convencional.

$H_1: \mu_1 \neq \mu_2$

b. Nivel de significancia, en condiciones normales se trabaja con un $\alpha = 0.05$

c. Estadístico de la prueba, prueba t para muestras independientes, utilizando la herramienta de IBM SPSS Statistics 22.

d. Prueba de Levene prueba de igualdad de varianza.

Hipótesis

H_0: Se asumen varianzas iguales

$H_0: \mu_1 = \mu_2$

H_1: No se asumen varianzas iguales

$H_1: \mu_1 \neq \mu_2$

Igualdad de varianza:

Sig. = 0.001

α= 5% = 0.05

Tabla 28. *Prueba de Levene, evaluación en olor*

Prueba de Levene			
		Prueba de Levene de calidad de varianzas	
		F	Sig.
PUNTUACIONES	Se asumen varianzas iguales	11,378	,001
	No se asumen varianzas iguales		

Conclusión

Como la significancia es menor que el alfa, aceptamos la hipótesis alternativa o de trabajo y rechazamos la hipótesis nula, por lo que **no se asumen varianzas iguales.**

e. Cálculo del estadístico de la prueba

En la siguiente tabla se muestra la media y la desviación estándar, donde se aprecia que hay una diferencia en los promedios de los productos, donde el **producto a** se muestra por encima del **producto b,** por lo que se asume que el producto a obtuvo mejores calificaciones en la evolución sensorial respecto al olor.

Tabla 29. *Estadístico de grupo, evaluación de olor*

Estadísticas de grupo					
	PRODUCTOS	N	Media	Desviación estándar	Media de error estándar
PUNTUACIONES	PRODUCTO A	50	84,200	19,88128	2,81164
	PRODUCTO B	50	80.000	30,30458	4,28571

En la siguiente tabla se muestra la significancia bilateral, el valor a utilizar es del segundo dato, porque no se asumen varianzas iguales.

Tabla 30. *Prueba de muestras independientes, prueba t para igualdad de medias*

Prueba t para igualdad de medias						
					95% de intervalo de confianza	
t	gl	Sig. (bilateral)	Diferencia de medias	Diferencia de error estándar	Inferior	Superior
,819	98	,415	4,20000	5,12569	-5,12569	14,37176
,819	84,587	,415	4,20000	5,12569	5,12569	14,39195

f. Decisión estadística

Significancia bilateral: 0.415

Valor de alfa: 0.05

Tenemos el valor de la significancia que es mayor que el valor del alfa, por lo tanto aceptamos la hipótesis nula por lo que se concluye que **no existe diferencia significativa** entre el olor del néctar elaborado con la asistencia del software utilizando información de calidad y la elaboración del néctar sin la utilización de software realizado de manera convencional.

4.6.4.3. Evaluación de sabor

a. Formulación de las hipótesis nula y alternativa

H_0: no existe diferencia significativa entre el sabor del néctar elaborado con la asistencia del software utilizando información de calidad y la elaboración del néctar sin la utilización de software realizado de manera convencional.

$H_0: \mu_1 = \mu_2$

H_1: existe diferencia significativa entre el sabor del néctar elaborado con la asistencia del software utilizando información de calidad y la elaboración del néctar sin la utilización de software realizado de manera convencional.

$H_1: \mu_1 \neq \mu_2$

b. Nivel de significancia, en condiciones normales se trabaja con un α = 0.05

c. Estadístico de la prueba, prueba t para muestras independientes, utilizando la herramienta de IBM SPSS Statistics 22.

d. Prueba de Levene, prueba de igualdad de varianza.

Hipótesis

H_0: se asumen varianzas iguales

H_0: $\mu_1 = \mu_2$

H_1: no se asumen varianzas iguales

H_1: $\mu_1 \neq \mu_2$

Igualdad de varianza:

Sig. = 0.094

α= 5% = 0.05

Tabla 31. *Prueba de Levene, evaluación en sabor*

Prueba de Levene			
		Prueba de Levene de calidad de varianzas	
		F	Sig.
PUNTUACIONES	Se asumen varianzas iguales	2,867	,094
	No se asumen varianzas iguales		

Conclusión

Como la significancia es mayor que el alfa, aceptamos la hipótesis nula, por lo que **se asumen varianzas iguales.**

e. Cálculo del estadístico de la prueba

En la siguiente tabla se muestra la media y la desviación estándar, donde se aprecia que hay una diferencia en los promedios de los productos, donde

el **producto a** se muestra por encima del **producto b,** por lo que se asume que el producto a obtuvo mejores calificaciones en la evolución sensorial respecto al sabor.

Tabla 32. *Estadístico de grupo, evaluación de sabor*

Estadísticas de grupo					
	PRODUCTOS	N	Media	desviación estándar	media de error estándar
PUNTUACIONES	PRODUCTO A	50	84,200	19,88128	2,81164
	PRODUCTO B	50	80.000	30,30458	4,,28571

En la siguiente tabla se muestra la significancia bilateral, el valor a utilizar es del primer dato, porque se asumen varianzas iguales.

Tabla 33. *Prueba de muestras independientes, prueba t para igualdad de medias*

Prueba t para igualdad de medias						
					95% de intervalo de confianza	
t	gl	Sig. (bilateral)	Diferencia de medias	Diferencia de error estándar	Inferior	Superior
,819	98	,415	4,20000	5,12569	-5,12569	14,37176
,819	84,587	,415	4,20000	5,12569	5,12569	14,39195

f. Decisión estadística

Significancia bilateral: 0.415

Valor de alfa: 0.05

Tenemos el valor de la significancia que es mayor que el valor del alfa, por lo tanto aceptamos la hipótesis nula por lo que se concluye que **no existe diferencia significativa** entre el sabor del néctar elaborado con la asistencia del software utilizando información de calidad y la elaboración del néctar sin la utilización de software realizado de manera convencional.

4.6.5. Gráficos de aceptación del software

Se realizó una encuesta general para evaluar la aceptación del software, realizado a 50 individuos, obteniendo los siguientes resultados:

1. Según sus ingresos mensuales ¿en qué rango se ubica?

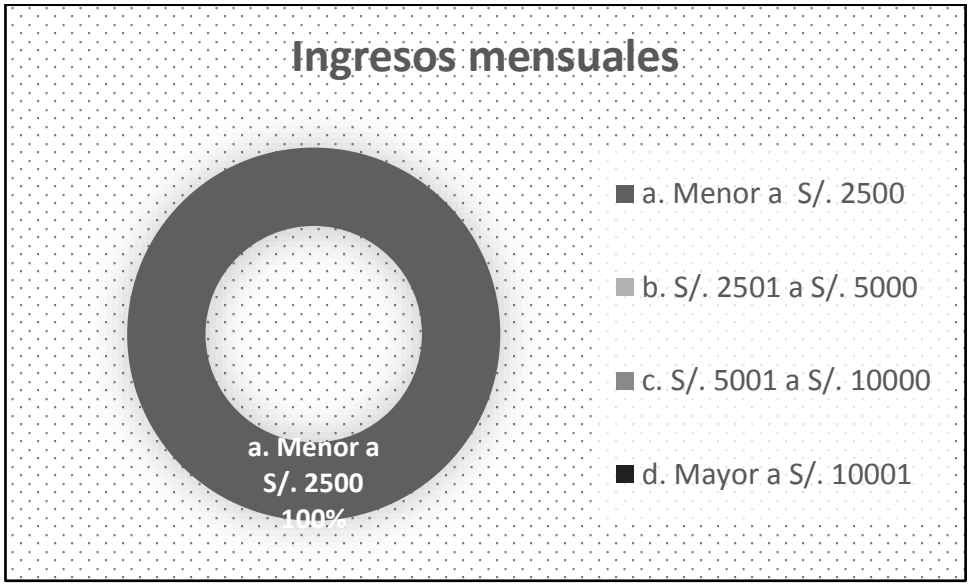

Figura 37. Ingresos mensuales de los individuos encuestados, se aprecia que la totalidad de encuestados tiene un ingreso menor a S/. 2500.

2. ¿Usa algún software en su trabajo o en su estudio?

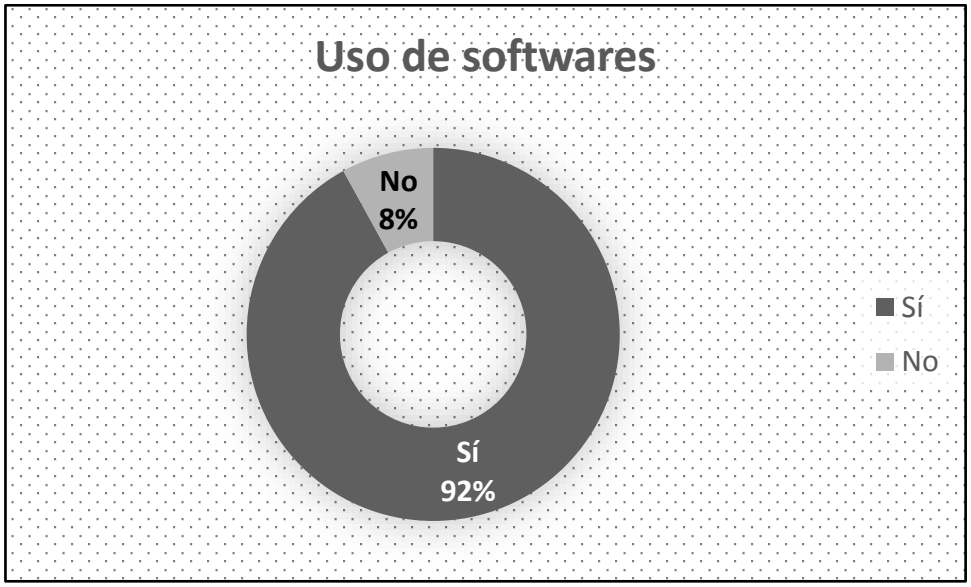

Figura 38. El 92% de encuestados mencionan que utilizan algún tipo de software dentro de su estudio o trabajo.

3. ¿Qué tipo de software usa con más frecuencia?

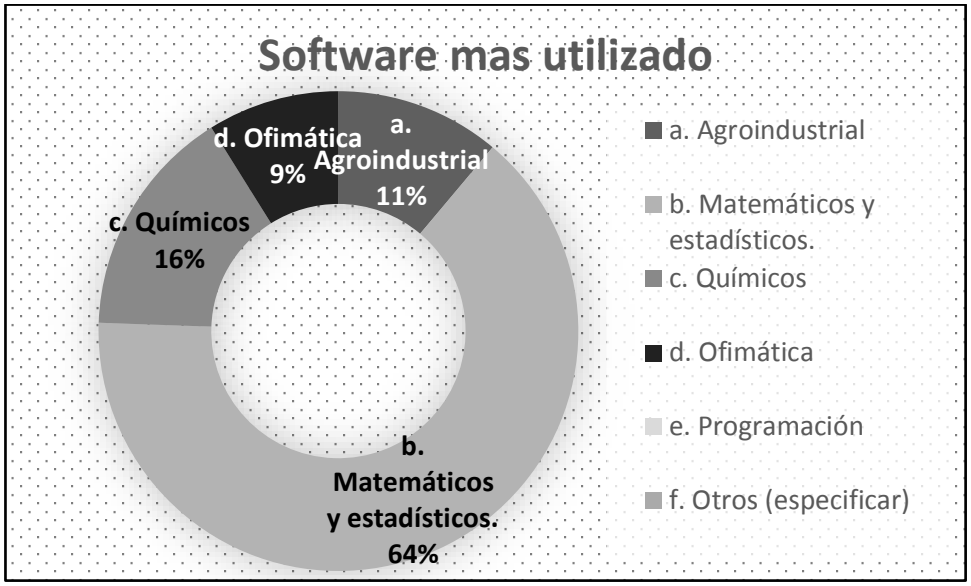

Figura 39. Se aprecia que la mayoría de los encuestados utilizan softwares matemáticos, donde solo un 11% utilizan softwares agroindustriales, debido a que este tipo de softwares no son muy conocidos.

4. ¿La arquitectura (diseño) del software, es de su agrado?

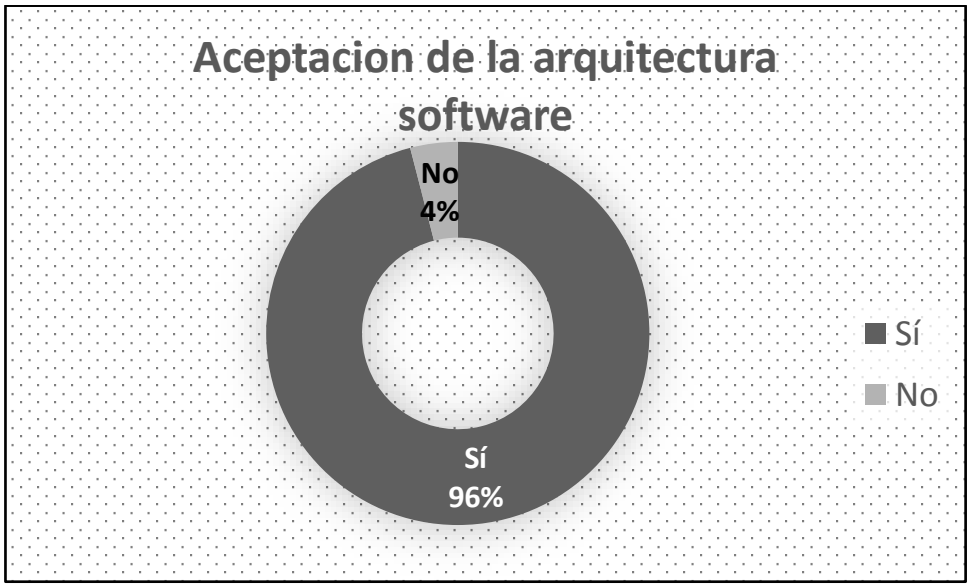

Figura 40. El 96% de los encuestados mencionaron que les agrada la arquitectura del software, lo que representa una aceptación muy buena.

5. ¿La funcionalidad del software de dosificar, formular y sistematizar el proceso le es eficiente?

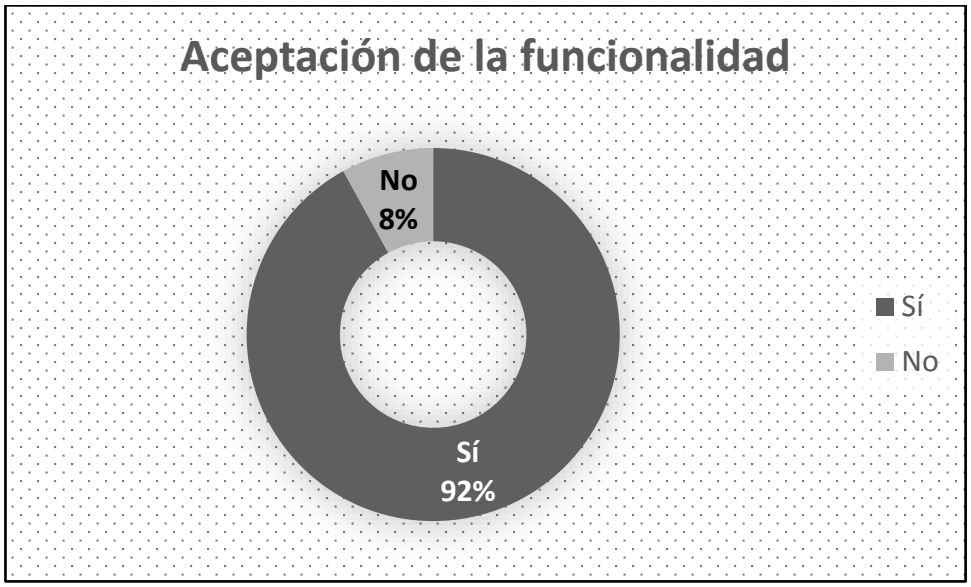

Figura 41. El 92% de los encuestados mencionaron que la funcionalidad del software es eficiente.

6. ¿Las características organolépticas del producto (néctar) dependen de la formulación de los insumos?

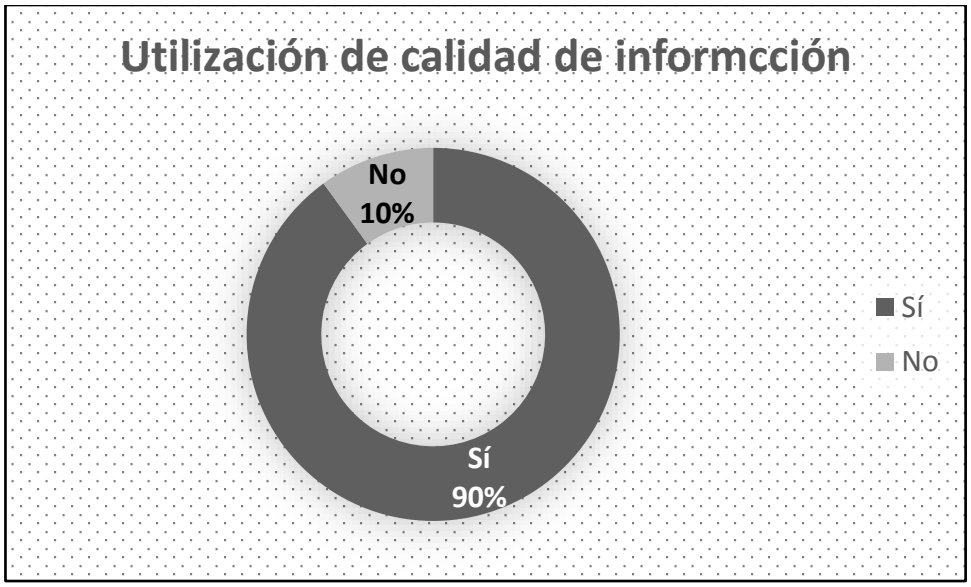

Figura 42. Tener un producto con buenas características organolépticas depende de la calidad de información que se utilice, así lo demuestran la mayoría de los encuestados.

7. ¿El software le es importante para el control del proceso?

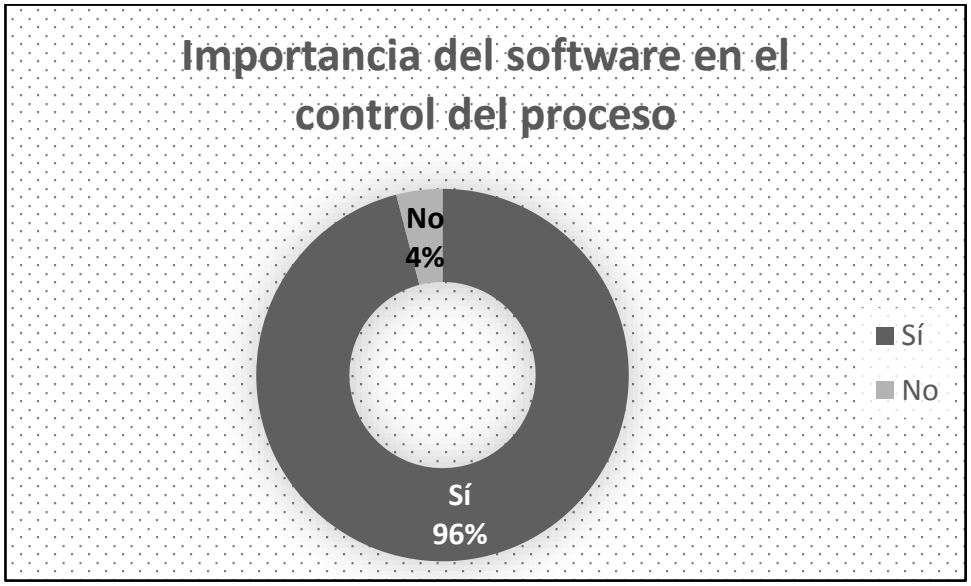

Figura 43. La mayoría afirma que el software desarrollado es importante para el control del proceso.

8. ¿Al revisar y usar el software, cree que le es útil?

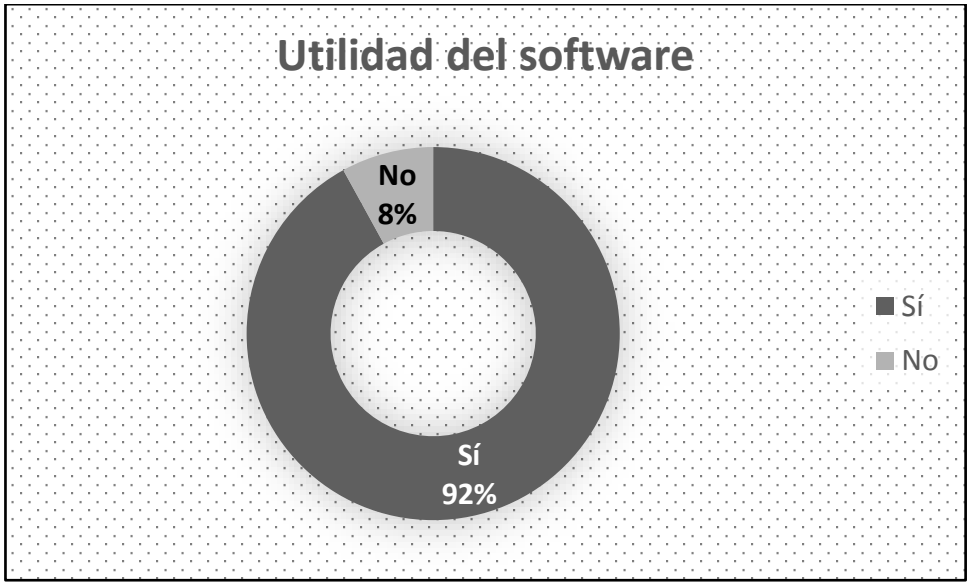

Figura 44. La mayoría afirma que el software desarrollado le es útil como herramienta para el manejo de información.

9. ¿Estaría dispuesto a adquirir el software?

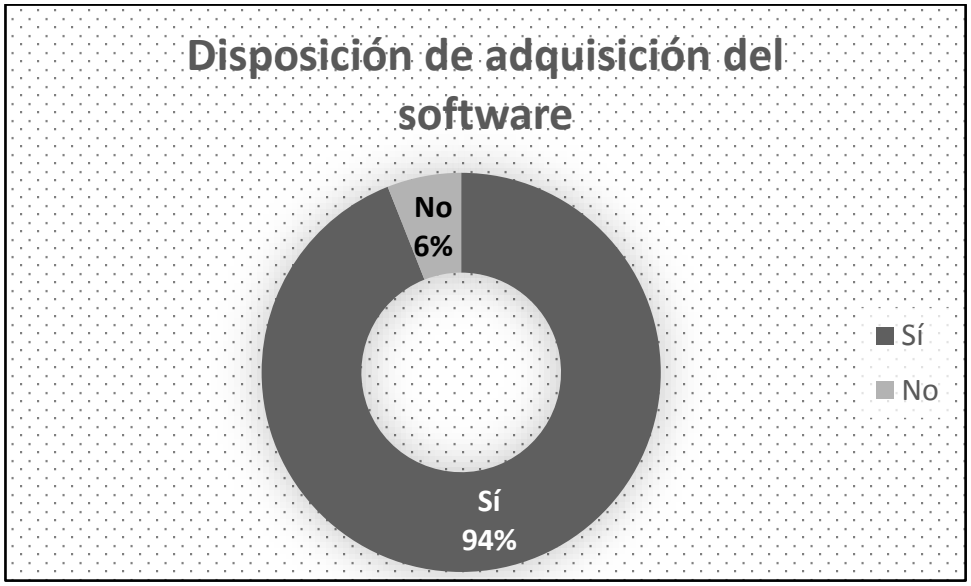

Figura 45. Un 94% están dispuestos a adquirir el software, por lo que representa una muy buena aceptación.

10. ¿Cuánto estaría dispuesto a pagar por el software?

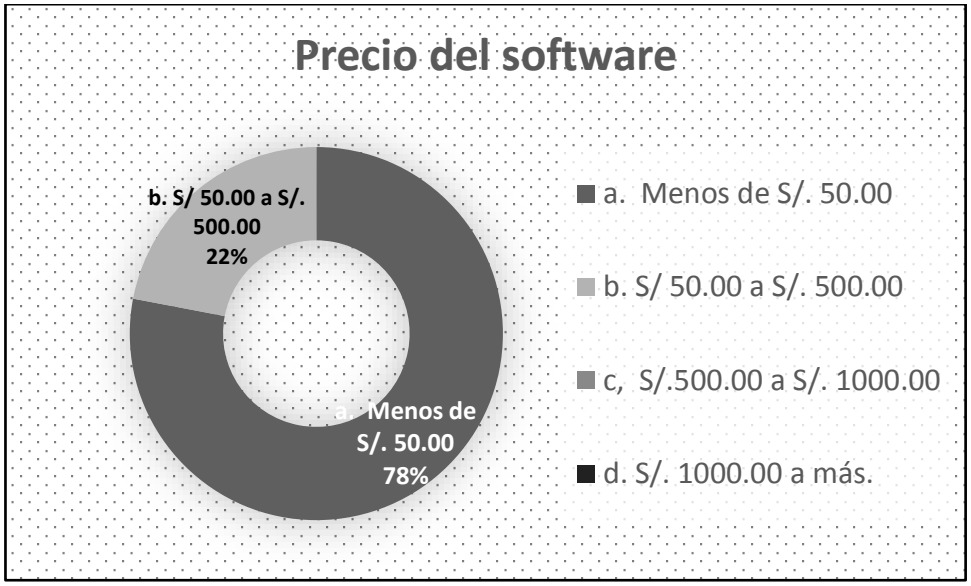

Figura 46. La mayoría estaría dispuesto a pagar un precio no mayor de S/. 50.00 por la compra del software.

V. DISCUSIÓN

5.1. Análisis de requisitos

El análisis y la especificación de requisitos se realizaron siguiendo los procedimientos de la ingeniería del software, obteniendo y laborando los documentos de Plan de iteración, Documento visión y Especificaciones de caso de uso, dando conformidad a lo que menciona Sawyer (1997), donde indica que se debe elaborar los documentos mencionados, para garantizar la ejecución del trabajo y agilizar la codificación de los algoritmos, para realizar la programación de una manera ordenada y eficiente; asimismo Martínez (2011) en su trabajo de investigación “Desarrollo de un software para la automatización de los procesos administrativos de la sección de almacén del núcleo Monagas de la universidad de oriente”, adopto los mismos procedimientos para el análisis de los requisitos en el software que desarrolló.

5.2. Diseño de la arquitectura

Realizar el diseño de la arquitectura del software es un papel muy importante dentro de la programación, y es una etapa que se realiza antes de comenzar a codificar el software, por lo que hemos realizado siguiendo el procedimiento especificado por Gottesdiener y Sawyer (2002), donde condicionan al programador seguir estrictos procedimientos, a lo cual adaptamos el diseño de la arquitectura del software; asimismo Darío (2009) en su investigación “Desarrollo de un software para el control de inventario de productos terminados para los departamentos de atención al cliente, la línea de producción “sector beta”, y despacho en una empresa alimentos”, realizo de manera similar el diseño de la arquitectura para su trabajo de investigación, obteniendo resultados similares.

5.3. Diseño y desarrollo del software para elaboración de néctar

El diseño y el desarrollo es la parte central de la programación, por lo que se ha seguido los pasos estructurados por Pressman (2003), en su libro “Ingeniería del Software, un enfoque Práctico”, llegando a codificar de acuerdo

a lo establecido; asimismo Darío (2009) en su investigación “Desarrollo de un software para el control de inventario de productos terminados para los departamentos de atención al cliente, la línea de producción “sector beta”, y despacho en una empresa alimentos”, logro realzar la codificación satisfactoriamente y parecido al presente trabajo.

5.4. Diseño y desarrollo del software para control del proceso

La automatización y el control del proceso se debe realizar siguiendo los parámetros de control y administración de datos, por lo que se realizó de acuerdo a Sanchis y Ariño (2010), con su libro “Automatización industrial” para la conexión de los dispositivos periféricos de los sensores, asimismo seguimos lo establecido por Sommerville (2005), en su libro “Ingeniería del software” donde especifica los procedimiento de codificación; a lo que obtuvimos el diseño profesional del software, mucho mejor de lo realizado por Loya (2010) en su investigación “Diseño e implementación de un programa de software para el dimensionamiento de un secador de doble tambor”, donde solo utilizo aspectos básicos de diseño y obtuvo el diseño y la interfaz gráfica muy básica.

5.5. Desarrollo del proceso de elaboración de néctar

La elaboración del néctar se realizó de acuerdo a las normas teniendo como referencia la Norma General para Zumos (jugos) y Néctares de Frutas (codex stan 247-2005), también con la consideración del Código Internacional Recomendado de Prácticas - Principios Generales de Higiene de los Alimentos (CAC/RCP 1-1969), el producto elaborado reúne similares condiciones a lo realizado por Aleman (2015) en su trabajo de investigación "Determinación de parámetros adecuados en la elaboración de un néctar tropical mixto de mango (Manguifera indica L) con ciruela (Spondias purpurea L)", donde utiliza un similar diagrama de flujo para el proceso de elaboración.

5.6. Determinación de las características fisicoquímicas y sensoriales del néctar y análisis estadístico

La determinación de las características fisicoquímicas realizado el laboratorio alimentario de la escuela profesional de ingeniería agroindustrial, de determino con los instrumentos implementados en este laboratorio, donde las características fisicoquímicas se encuentran dentro del rango establecido por la Norma General ara Zumos (Jugos) y Néctares de Frutas (Codex Stan 247-2005), asimismo estos resultados son similares a Aleman (2015) en su trabajo de investigación "Determinación de parámetros adecuados en la elaboración de un néctar tropical mixto de mango (Manguifera indica L) con ciruela (Spondias purpurea L)", obtiene similares resultados.

VI. CONCLUSIONES

Al término de la investigación concluimos:

1. Se concluye de acuerdo a nuestro objetivo principal que el CAM (Manufactura Asistida por Computadora) a través del desarrollo de software, mediante la prueba Chi-cuadrado prueba de aceptación que si influye en la eficiencia en el proceso de elaboración de néctar.

2. Se diseñó y desarrollo el software al 95% con estándares de calidad en codificación por lo que el software desarrollado contribuye eficientemente al 100% en la dosificación correcta de elaboración de néctar, permitiendo a través de una arquitectura avanzada de software el ingreso dinámico y fluido de datos.

3. Se desarrolló el software basado en algoritmos con los más altos estándares de calidad en diseño y codificación que realicen eficientemente el control del proceso de elaboración de néctar, mediante la estructuración de información de calidad de acuerdo a las normas de procesamiento de productos alimentarios.

4. Se determinó las características fisicoquímicas y sensoriales del néctar elaborado con la asistencia del software, obteniendo los resultados de acuerdo a la norma del CODEX STAN 247-2005, donde presenta un pH de 4.2 y 14 °Brix, encontrándose dentro del rango aceptado por las normas y teniendo una aceptación por los consumidores; con una evaluación mediante la escala (tabla 24), se obtuvo una calificación de EXCELENTE respecto al color, olor y sabor, esta calificación indica una muy buena aceptación del producto.

5. Concluimos respecto al análisis estadístico, según la prueba Chi-cuadrado el software desarrollado si tiene aprobación de los usuarios, según la prueba Chi-cuadrado de asociación u homogeneidad el

desarrollo del software con calidad de información si influirá en el proceso de elaboración del néctar y según la prueba T de muestras independientes, existe diferencia significativa entre el color del néctar elaborado con la asistencia del software y la elaboración del néctar sin la utilización de software realizado de manera convencional; no existe diferencia significativa entre el olor del néctar elaborado con la asistencia del software y la elaboración del néctar sin la utilización de software y no existe diferencia significativa entre el sabor del néctar elaborado con la asistencia del software y la elaboración del néctar sin la utilización de software.

VII. RECOMENDACIONES

1. Para realizar un análisis de requisitos de software, se recomienda a los estudiantes, analistas y programadores, recopilar los datos directamente desde el campo de estudio, y no indirectamente mediante revisiones documentales, los documentos solo deben ser utilizados para contrastar y verificar la información.

2. Para el diseño de la arquitectura del software, recomendamos a los estudiantes, analistas y programadores, que se debe realizar enfocado a la arquitectura de Windows 8 o superior, y la interfaz gráfica debe ser de acuerdo a la información a mostrar y de acuerdo a la adaptabilidad del usuario.

3. Para la construcción del software, se recomienda a los estudiantes, analistas y programadores, que se debe realizar con lenguajes de programación que brinden una alta seguridad para la codificación asegurando el éxito del proyecto, asimismo utilizar entornos de desarrollo que brinden además de la codificación una visibilidad del avance respecto a los objetos utilizados en el software.

4. Para el desarrollo del proceso de elaboración de néctar, recomendamos a los estudiantes de las carreras relacionadas a los alimentos, realizar este proceso de acuerdo a las normas vigentes, cumpliendo estrictamente la calidad higiénico-sanitaria del producto.

5. Recomendamos a todas las empresas y micro empresas implementar tecnologías en las líneas de producción de la empresa como el uso del software, que permite estructurar y organizar la información y el control de la operación del proceso de producción.

6. Recomendamos a todas las universidades y empresas, implementar las tecnologías de última generación, porque hoy en día, las aplicaciones y

diversos tipos de software a la medida se han convertido en la base tecnológica de las empresas modernas. Sin embargo, como toda nueva herramienta, el diseño de la estrategia así como la estandarización de procesos que permitan aprovechar al máximo estas modernas tecnologías dependen de la asesoría de un consultor experto en el desarrollo de este tipo de sistemas. Desde actualizar su imagen corporativa para hacerla atractiva e impactante, generar contenidos inteligentes hasta el desarrollo de sistemas y software para empresas. Invertir en sistemas y software para empresas es invertir en eficiencia, ya que los beneficios que puede tener con este tipo de tecnologías digitales pueden no sólo mejorar sus procesos sino incrementar el desarrollo y los alcances de su empresa. Si se tiene un negocio referente a productos, un sistema puede no sólo auto administrar la atención al cliente sino también la parte de logística referente a inventarios, pedidos y envío de mercancía. A partir de un registro de usuarios, el sistema genera una base de datos que le brinda el beneficio de saber cuántos clientes tiene hasta la fecha, definiendo en cada uno de ellos su perfil, como su historial de compra (para el manejo de descuentos por ser cliente frecuente) o datos que pueden ser de utilidad (cumpleaños, aniversario) para generar una buena relación cliente-empresa, enviando correos electrónicos a las cuentas de los usuarios para felicitarlos, enviarles la oferta del mes o para resolver sus dudas o contestar sus comentarios. Un sistema permite crear una comunidad donde puede tener a sus clientes frecuentes en una sección VIP, por ejemplo, o crear un foro o Chat que responda las preguntas más frecuentes que tienen los usuarios sobre sus productos, siendo este tipo de secciones ideales para anunciar el lanzamiento de una nueva línea de artículos o noticias relacionadas a su negocio. Después de analizar, se crea un estándar de reglas que debe cumplir o llevar un software, en general se desarrolla para usted la plataforma ideal para su negocio, ya sea desarrollo de mapas geográficos, sistema de reservaciones on line, sistemas de gestión, inventarios en línea, renta de tienda en línea, registro de usuarios, posicionamiento web (SEP, SEO).

VIII. LITERATURA CITADA

1. Alemán C. (2015). *"Determinación de parámetros adecuados en la elaboración de un néctar tropical mixto de mango (Manguifera indica L) con ciruela (Spondias purpurea L)".* Tesis de Ingeniero Agroindustrial e Industrias Alimentarias. Facultad de Ingeniería de Industrial, Universidad Nacional de Piura. Piura, Perú. 45 p.

2. Banzi, Massimo (2012). *Introducción a arduino.* Anaya Multimedia. ISBN 8441531773, 9788441531772. 78 p.

3. Chase, B. (2009). *Administración de operaciones, producción y cadena de suministros (Duodécima edición).* México, McGRAW-HILL. ISBN: 978-970-10-7027-7. 56 p.

4. Darío, R. (2009). *Desarrollo de un software para el control de inventario de productos terminados para los departamentos de atención al cliente, la línea de producción "sector beta", y despacho en una empresa alimentos.* Tesis de Ingeniero en Computación. Departamento de Ingeniería y Ciencias Aplicadas, Universidad de Oriente. Barcelona, España. 66 p.

5. Fowler, Martin; Kendall Sccott (1999). *UMLGota a Gota*. Addison Wesley. ISBN 9789684443648. 103 p.

6. Gaither, N.; Frazier, G. (2006). *Administración de producción y operaciones* (octava edición). EE.UU. ISBN 88-481-3214-7. 88 p.

7. Gottesdiener, Ellen; P. Sawyer (2002). *Requirements by Collaboration: Workshops for Defining Needs* (en inglés). Addison-Wesley Professional. ISBN 978-0201786064. 112 p.

8. Haeberer, A.; Veloso, G.; Baum, G. (1988). *Formalización del proceso de desarrollo de software* (Ed. preliminar edición). Buenos Aires: Kapelusz. ISBN 950-13-9880-3. 162 p.

9. Jacobson, B; Rumbaugh. B. (1999). *UML - El Lenguaje Unificado de Modelado*. Pearson Addisson-Wesley. Rational Software Corporation, Addison Wesley Iberoamericana. ISBN 84-7829-028-1. 176 p.

10. Jacobson, I.; Booch, G.; Rumbaugh, J. (2000). *El Proceso Unificado de Desarrollo de Software*. Pearson Addisson-Wesley. 226 p.

11. Kalpakjian, S.; Schmid, S. (2008). *Manufactura, ingeniería y tecnología (*Quinta edición). México, ISBN: 978-970-26-1026-7. 155 p.

12. Loucopoulos, Pericles; Karakostas, V. (1995). *System Requirements Engineering* (en inglés). London: McGraw-Hill Companies. ISBN 978-0077078430. 133 p.

13. Loya, P. (2010). *Diseño e implementación de un programa de software para el dimensionamiento de un secador de doble tambor*. Tesis de Ingeniería Química. Facultad de Ingeniería Química y Agroindustria, Escuela Politécnica Nacional. Quito, Ecuador. 45 p.

14. Martínez, Pedro (2011). *Desarrollo de un software para la automatización de los procesos administrativos de la sección de almacén del núcleo Monagas de la universidad de oriente*. Tesis de ingeniero de sistemas. Ingeniería de sistemas, Universidad de Oriente. Monagas, Venezuela. 61 p.

15. Mendoza, C. (2008). *Enfoques conceptuales de la Metodología RUP para el desarrollo de Software.* Publicaciones de la Universidad de la Sabana. Bogotá, Colombia. 54 p.

16. Pilligua, I. (2006). *Diseño de un Software para calcular cámaras frigoríficas.* Título de ingeniero mecanico. Facultad de Ingeniería en Mecánica y Ciencias de la Producción, Escuela Superior Politécnica Del Litoral. Guayaquil, Ecuador. 66 p.

17. Pressman, R. (2003). *Ingeniería del Software, un enfoque Práctico* (Quinta edición edición). Mc Graw Hill. ISBN 84-481-3214-9. 98 p.

18. Sanchis, R.; Romero, J.; Ariño, C. (2010). *Automatización industrial* (Primera edición). ISBN: 978-84-693-0994-0. 245 p.

19. Sommerville, Ian (2005). *Ingeniería del software* (7ma. edición). Madrid: Pearson Educación S.A. ISBN 84-7829-074-5. 221 p.

20. Sommerville, Ian; P. Sawyer (1997). *Requirements Engineering: A Good Practice Guide* (en inglés) (1ra. edition edición). Wiley&Sons. ISBN 978-0471974444. 332 p.

ANEXOS

ANEXO Nº 1: Diagrama de flujo del proceso de desarrollo del software.

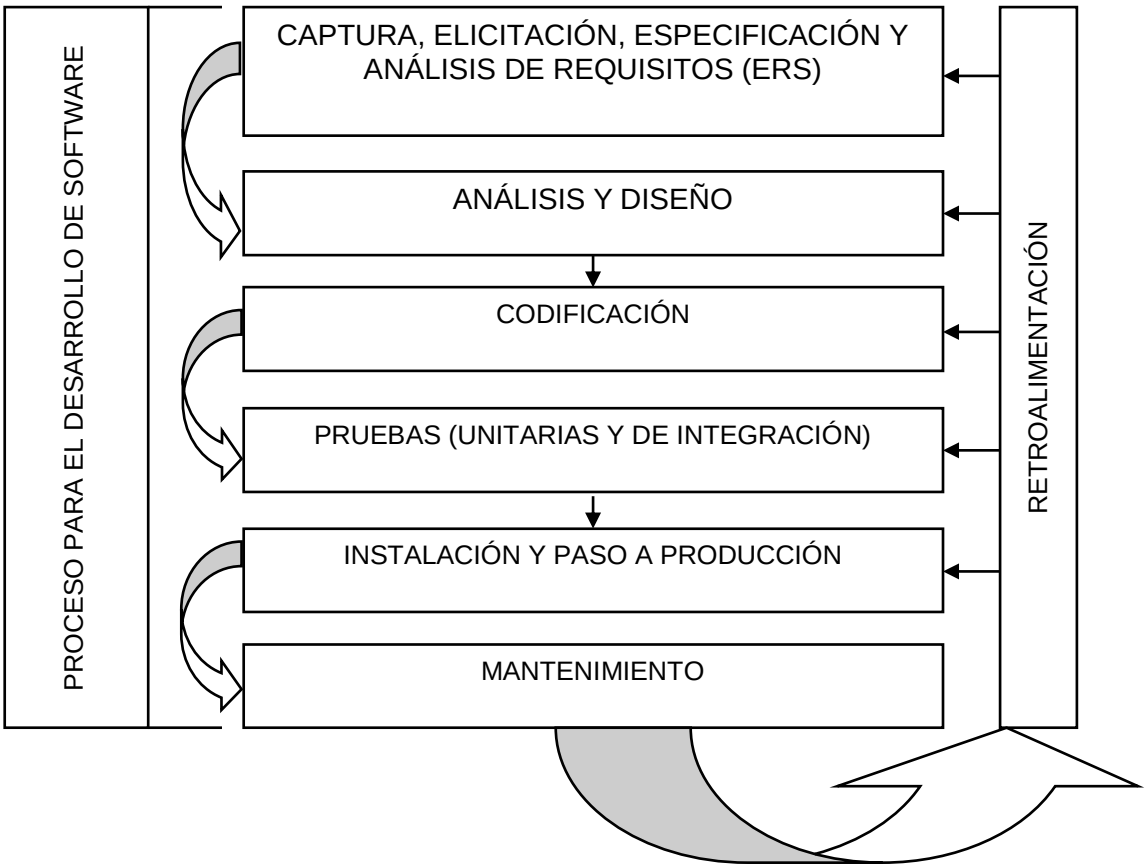

Figura 47. Diagrama de flujo del proceso de desarrollo del software.

ANEXO Nº 2: Documento glosario

Proyecto: “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Nombre del documento:	Versión:	Fecha:
Documento visión	1.0	Agosto 2016

Proyecto:
“Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Documento visión.
Versión 1.0

Proyecto: “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Nombre del documento:	Versión:	Fecha:
Documento visión	1.0	Agosto 2016

Glosario

Introducción

En este documento se definen los términos más importantes que se utilizaran en la realización del proyecto “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”.

Propósito

El presente documento tiene como propósito dar las definiciones exactas y sin ambigüedades de la terminología utilizada en el proyecto “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”.

Alcance

El Presente documento se extiende a todo el subsistema del proyecto “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”.

Referencias

El presente glosario hace referencia a los siguientes documentos:

- Documento Visión.
- Plan de iteración.

Organización del Glosario

El presente documento está organizado por definiciones de términos ordenados alfabéticamente.

Proyecto: “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”

Nombre del documento:	Versión:	Fecha:
Documento visión	1.0	Agosto 2016

Definiciones

A continuación se presentan todos los términos manejados a lo largo de todo el proyecto de “Influencia de la manufactura asistida por computadora (CAM) en la eficiencia del proceso de elaboración de néctar”.

Actor:

Un actor es aquel que el rol o función que asume una persona, sistema o entidad que interactúa con el sistema que estamos construyendo de la misma forma. Tiene la propiedad de ser externo al sistema. Hay que tener en cuenta que un usuario puede acceder al sistema como distintos actores.

Casos de Uso:

Es una técnica para capturar requisitos potenciales de un nuevo sistema o una actualización de software. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico.

Mercancía: producto del trabajo destinado a satisfacer alguna necesidad del hombre y que se elabora para la venta, no para el propio consumo.

Proveedor: persona o empresa que abastece de algunos artículos necesarios.

Sistema Automatizado: sistema donde se transfieren las tareas de producción, realizadas habitualmente por operadores humanos a un conjunto de elementos tecnológicos.
Compras: adquisición de materiales, bienes o servicios mediante el pago de dinero.

Producto: Bien o servicio que se obtiene con la ejecución de un proyecto.

Proceso: Acción o sucesión de acciones continuas regulares, que ocurren o se llevan a cabo de una forma definida, y que llevan al cumplimiento de algún resultado; una operación continua o una serie de operaciones.

Proyecto: conjunto de acciones que conducen a un fin determinado. Es un conjunto de actividades concretas, interrelacionadas y coordinadas entre sí, que se realizan a fin de producir determinados bienes y servicios capaces de satisfacer necesidades o resolver problemas.

Sistema: Conjunto de procesos o elementos interrelacionados con un medio para formar una totalidad encauzada hacia un objetivo común.

Software: Se denomina software, programa, equipamiento lógico o soporte lógico a todos los componentes intangibles de una computadora, es decir, al conjunto de programas y procedimientos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema (hardware).

ANEXO Nº 3: Matriz de consistencia

Tabla 34. *Matriz de consistencia*

"INFLUENCIA DE LA MANUFACTURA ASISTIDA POR COMPUTADORA (CAM) EN LA EFICIENCIA DEL PROCESO DE ELABORACIÓN DE NÉCTAR"				
PROBLEMA	**OBJETIVO**	**HIPÓTESIS**	**VARIABLE**	**INDICADORES**
PROBLEMA GENERAL:	**OBJETIVO GENERAL:**	**HIPÓTESIS GENERAL:**	**VARIABLE INDEPENDIENTE**	- Programación - Codificación - Arquitectura del software - Visual Basic .NET - NetFrameworks 4.5
¿De qué manera el CAM (Manufactura Asistida por Computadora) influirá en la eficiencia del proceso de elaboración de néctar?	Determinar que el CAM (Manufactura Asistida por Computadora) a través del desarrollo de software influye en la eficiencia del proceso de elaboración de néctar.	Si diseñamos y desarrollamos un software de calidad (CAM) entonces su aplicación influirá en la eficiencia del proceso de elaboración de néctar.	CAM - Manufactura asistida por computadora	
PROBLEMAS ESPECÍFICOS:	**OBJETIVOS ESPECÍFICOS:**	**HIPÓTESIS ESPECÍFICAS:**	**VARIABLES DEPENDIENTES**	- Fruta (gr.) - Agua (gr.) - Azúcar (gr.) - Espesante (gr.) - Conservante (gr.) - Corrector de pH (gr.)
¿De qué manera el desarrollo del software contribuirá en la eficiencia de la formulación de insumos para la elaboración de néctar?	Diseñar y desarrollar un software con estándares de calidad en codificación, para que contribuya en la eficiencia para la dosificación correcta de elaboración de néctar.	Si desarrollamos el software con estándares de calidad en la codificación, entonces su aplicación influirá eficientemente en la formulación de la elaboración de néctar.	Formulación	
¿De qué manera el desarrollo del software influirá en la eficiencia del control del proceso de elaboración de néctar?	Desarrollar un software que influya eficientemente en el control del proceso de elaboración de néctar.	El desarrollo del software influirá en la eficiencia del control del proceso de elaboración de néctar a mediante la sistematización de la información.	Proceso	- Temperatura (ºC) - Tiempo (min)
¿Cuál será las características fisicoquímicas y sensoriales del néctar elaborado, al usar el software como herramienta en el proceso de elaboración?	Determinar las características fisicoquímicas y sensoriales del néctar elaborado con la asistencia del software.	Las características fisicoquímicas y sensoriales del néctar elaborado, al usar el software como herramienta en el proceso de elaboración serán aceptable por los consumidores.	Características fisicoquímicas	- ºBrix - pH
			Características sensoriales	- Color - Olor - Sabor

ANEXO Nº 4: Formatos de análisis estadístico

EVALUACIÓN DE LA CALIDAD DE SOFTWARE

I. DATOS DE LA PERSONA ENTREVISTADA:

NOMBRE(S):……
PROFESIÓN:……
CENTRO DE TRABAJO:……FECHA: ……

II. INTRODUCCIÓN

La presente encuesta es parte de un trabajo de investigación; el cual servirá para determinar el grado de aceptación del software que tiene la función de dosificar los cálculos de insumos, descripción del proceso y balance de materia del proceso productivo de néctar. Se agradece su colaboración.

III. OBJETIVOS

Determinar el nivel de aceptabilidad del software en la ciudad de Huánuco.

IV. EVALUACIÓN

Califique el software en una escala valorativa de EXCELENTE a DEFICIENTE, considerando de la más alta calidad hasta la calidad más deficiente.

Escala / Características	EXCELENTE	MUY BUENO	BUENO	REGULAR	DEFICIENTE
FUNCIONALIDAD					
FIABILIDAD					
USABILIDAD					
EFICIENCIA					
MANTENIBILIDAD					
PORTABILIDAD					
CALIDAD EN USO					

4.1. RESUMEN

Funcionalidad - Un conjunto de atributos que se relacionan con la existencia de un conjunto de funciones y sus propiedades específicas. Las funciones son aquellas que satisfacen las necesidades implícitas o explícitas.

- Adecuación - Atributos del software relacionados con la presencia y aptitud de un conjunto de funciones para tareas especificadas.
- Exactitud - Atributos del software relacionados con la disposición de resultados o efectos correctos o acordados.
- Interoperabilidad - Atributos del software que se relacionan con su habilidad para la interacción con sistemas especificados.
- Seguridad - Atributos del software relacionados con su habilidad para prevenir acceso no autorizado ya sea accidental o deliberado, a programas y datos.
- Cumplimiento funcional.

Fiabilidad - Un conjunto de atributos relacionados con la capacidad del software de mantener su nivel de prestación bajo condiciones establecidas durante un período establecido.

- Madurez - Atributos del software que se relacionan con la frecuencia de falla por fallas en el software.
- Recuperabilidad - Atributos del software que se relacionan con la capacidad para restablecer su nivel de desempeño y recuperar los datos directamente afectos en caso de falla y en el tiempo y esfuerzo relacionado para ello.
- Tolerancia a fallos - Atributos del software que se relacionan con su habilidad para mantener un nivel especificado de desempeño en casos de fallas de software o de una infracción a su interfaz especificada.
- Cumplimiento de Fiabilidad - La capacidad del producto software para adherirse a normas, convenciones o legislación relacionadas con la fiabilidad.

Usabilidad - Un conjunto de atributos relacionados con el esfuerzo necesario para su uso, y en la valoración individual de tal uso, por un establecido o implicado conjunto de usuarios.

- Aprendizaje- Atributos del software que se relacionan al esfuerzo de los usuarios para reconocer el concepto lógico y sus aplicaciones.

- Comprensión - Atributos del software que se relacionan al esfuerzo de los usuarios para reconocer el concepto lógico y sus aplicaciones.
- Operatividad - Atributos del software que se relacionan con el esfuerzo de los usuarios para la operación y control del software.
- Atractividad

Eficiencia - Conjunto de atributos relacionados con la relación entre el nivel de desempeño del software y la cantidad de recursos necesitados bajo condiciones establecidas.

- Comportamiento en el tiempo - Atributos del software que se relacionan con los tiempos de respuesta y procesamiento y en las tasas de rendimientos en desempeñar su función.
- Comportamiento de recursos - Usar las cantidades y tipos de recursos adecuados cuando el software lleva a cabo su función bajo condiciones determinadas.

Mantenibilidad - Conjunto de atributos relacionados con la facilidad de extender, modificar o corregir errores en un sistema software.

- Estabilidad - Atributos del software relacionados con el riesgo de efectos inesperados por modificaciones.
- Facilidad de análisis - Atributos del software relacionados con el esfuerzo necesario para el diagnóstico de deficiencias o causas de fallos, o identificaciones de partes a modificar.
- Facilidad de cambio - Atributos del software relacionados con el esfuerzo necesario para la modificación, corrección de falla, o cambio de ambiente.
- Facilidad de pruebas - Atributos del software relacionados con el esfuerzo necesario para validar el software modificado.

Portabilidad - Conjunto de atributos relacionados con la capacidad de un sistema de software para ser transferido y adaptado desde una plataforma a otra.

- Capacidad de instalación - Atributos del software relacionados con el esfuerzo necesario para instalar el software en un ambiente especificado.

- Capacidad de reemplazamiento - Atributos del software relacionados con la oportunidad y esfuerzo de usar el software en lugar de otro software especificado en el ambiente de dicho software especificado.

  Calidad en uso - Conjunto de atributos relacionados con la aceptación por parte del usuario final y Seguridad.
- Eficacia - Atributos relacionados con la eficacia del software cuando el usuario final realiza los procesos.
- Productividad - Atributos relacionados con el rendimiento en las tareas cotidiana realizadas por el usuario final.
- Seguridad - Atributos para medir los niveles de riesgo.
- Satisfacción - Atributos relacionados con la satisfacción de uso del software

_____	_____
Firma del encuestado(a)	***Firma del encuestador***
DNI:	***DNI:***

EVALUACIÓN DE SOFTWARE

II. OBJETIVOS:

Determinar el nivel de aceptabilidad del software desarrollado en la ciudad de Huánuco.

II. INTRODUCCIÓN

La presente encuesta es parte de un trabajo de investigación (tesis); el cual servirá para determinar el grado de aceptación del software y para determinar si el valor de la variable de las características organolépticas del néctar depende de la variable de la formulación de insumos y la calidad de información. Se agradece su colaboración.

III. DATOS DEL ENTREVISTADO (A)

NOMBRE(S):……

OCUPACIÓN:……..……..

CENTRO DE TRABAJO:……FECHA: ……..

IV. PREGUNTAS

Marque con una X o encierre con un círculo, según crea conveniente.

1. *Según sus ingresos mensuales ¿en qué rango se ubica?*

a. Menos de S/. 2500.00
b. S/ 2500.00 a S/. 5000.00
c. S/.5000.00 a S/. 10000.00
d. S/. 10000.00 a más.

2. *¿Usa algún software en su trabajo o en su estudio?*

☐ *Sí*

☐ *No*

Si la respuesta es NO pasar a la pregunta Nº 4.

3. *¿Qué tipo de software usa con más frecuencia?*

a. Agroindustrial
b. Matemáticos y estadísticos.
c. Químicos
d. Ofimática
e. Programación
f. Otros (especificar)

……

4. *¿La arquitectura (diseño) del software, es de su agrado?*

☐ *Si*

☐ *No*

Sugerencia……

5. *¿La funcionalidad del software de dosificar, formular y sistematizar el proceso le es eficiente?*

☐ *Si*

☐ *No*

Sugerencia……

6. *¿Las características organolépticas del producto (néctar) dependen de la formulación de los insumos?*

☐ *Si*

☐ *No*

Sugerencia……

7. *¿El software le es importante para el control del proceso?*

☐ *Si*

☐ *No*

Sugerencia……

8. *¿Al revisar y usar el software, cree que le es útil?*

☐ *Si*

☐ *No*

Sugerencia……

9. *¿Estaría dispuesto a adquirir el software?*

☐ *Si*

☐ *No*

Sugerencia……

10. *¿Cuánto estaría dispuesto a pagar por el software?*

a) *Menos de S/. 50.00*
b) *S/ 50.00 a S/. 500.00*
c) *S/.500.00 a S/. 1000.00*
d) *S/. 1000.00 a más.*

11. *Recomendaciones u observaciones*

.....
.....
.....

EVALUACIÓN DE DOS MUESTRAS DIFERENTES

I. DATOS DE LA PERSONA ENTREVISTADA:

OCUPACIÓN:……

CENTRO DE TRABAJO:……FECHA: ……

II. OBJETIVOS

Determinar las diferencias entre dos muestras.

III. INTRODUCCIÓN

La presente encuesta es parte de un trabajo de investigación, con la cual se pretende determinar las diferencias entre dos muestras, respecto a las características organolépticas, que fueron elaboradas en condiciones diferentes.

IV. EVALUACIÓN

Califique el producto marcando con una [X] en el recuadro correspondiente a la escala valorativa de EXCELENTE a DEFICIENTE, considerando de la más alta calidad hasta la calidad más deficiente.

ESCALA DE CATEGORÍAS DE VALORACIÓN

PRODUCTO A					
ESCALA / CRTCAS.	EXCELENTE	MUY BUENO	BUENO	REGULAR	DEFICIENTE
COLOR					
OLOR					
SABOR					

PRODUCTO B					
ESCALA / CRTCAS.	EXCELENTE	MUY BUENO	BUENO	REGULAR	DEFICIENTE
COLOR					
OLOR					
SABOR					

V. RECOMENDACIONES U OBSERVACIONES

……
……
……
……

ANEXO Nº 5: Panel fotográfico

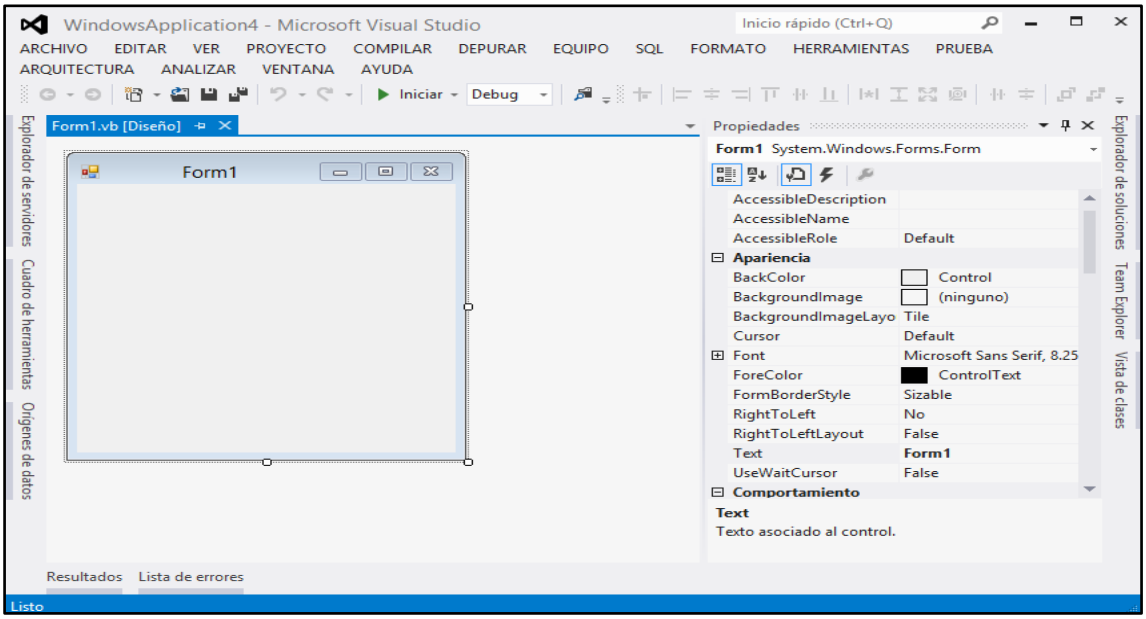

Figura 48. Entorno de desarrollo.

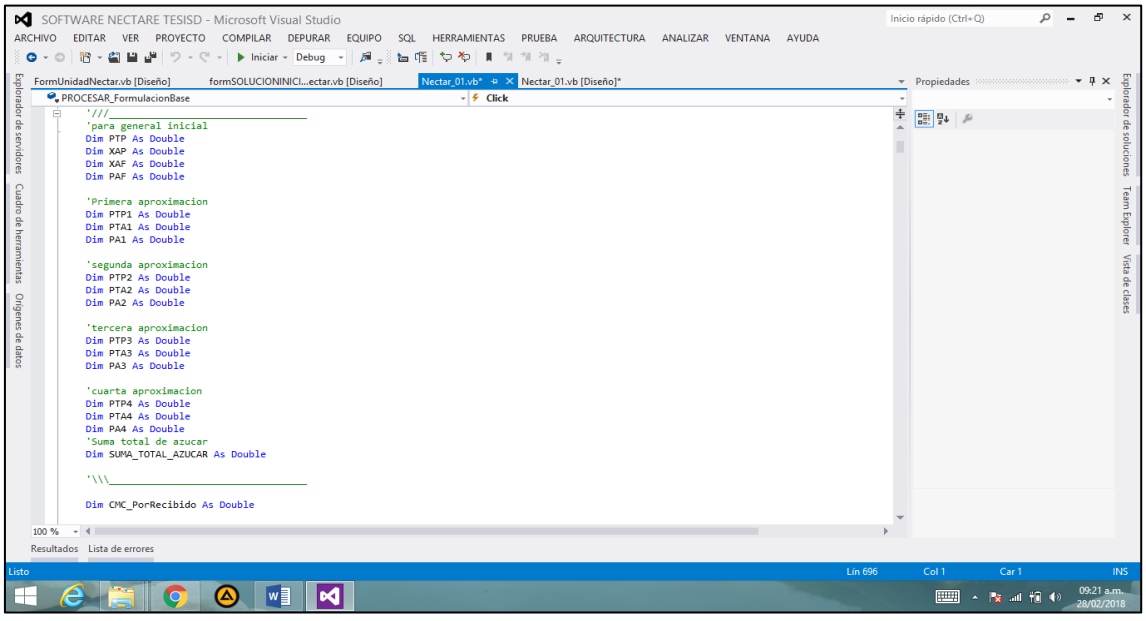

Figura 49. Declaración de variables.

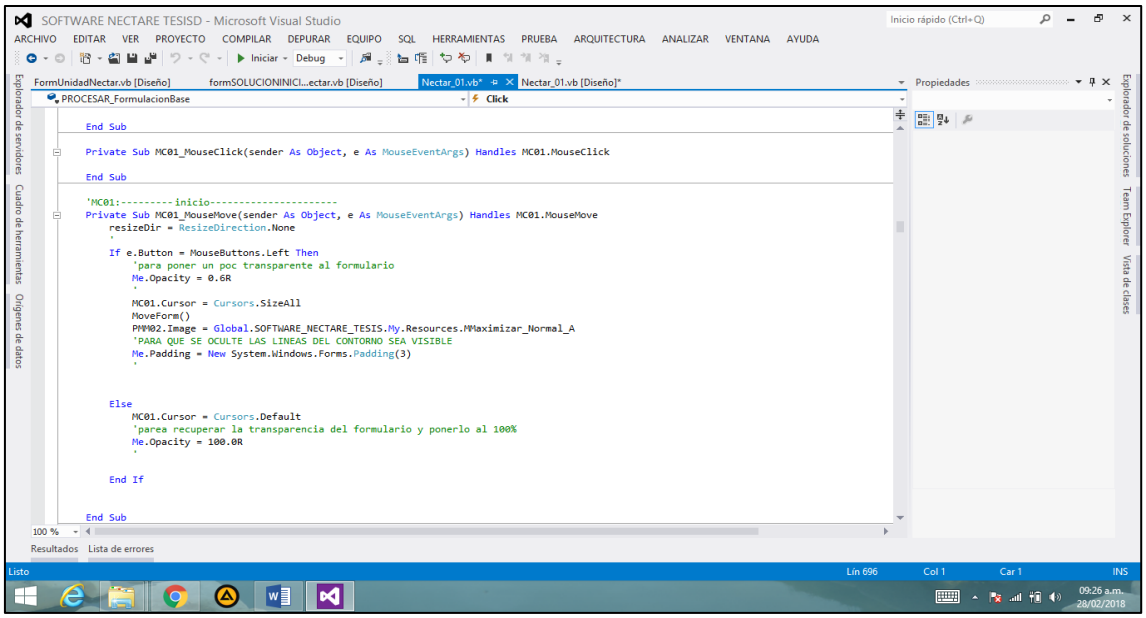

Figura 50. Inicio de la objetos.

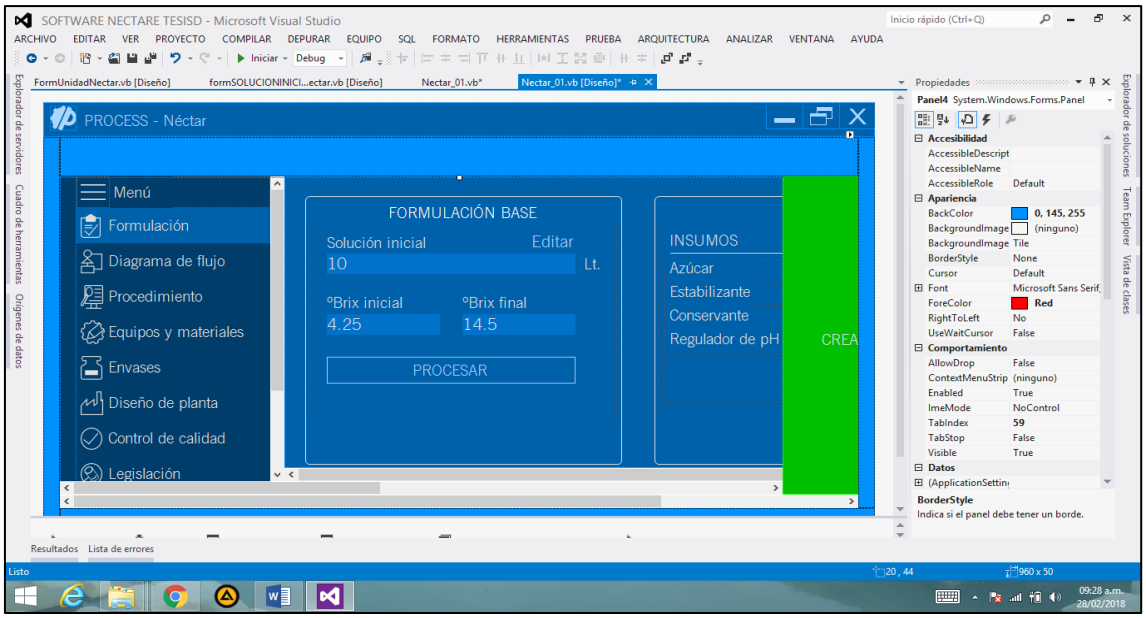

Figura 51. Arquitectura del software.

Figura 52. Materiales y equipos.

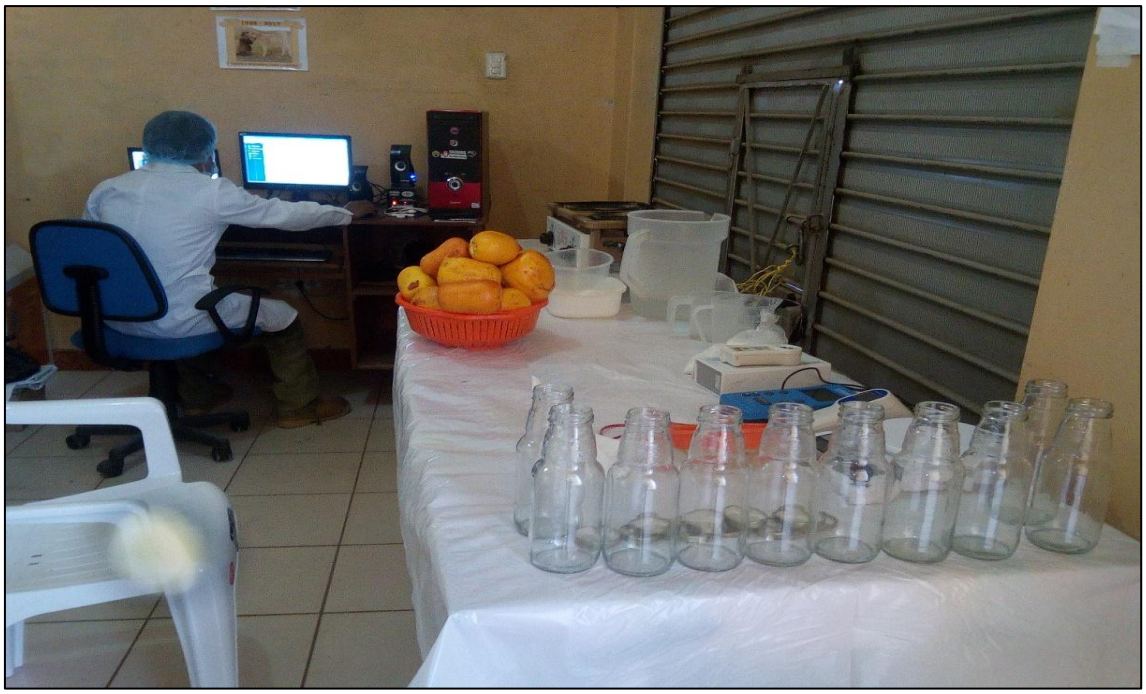

Figura 53. Asistencia del software en el proceso.

Figura 54. Licuado de materia prima.

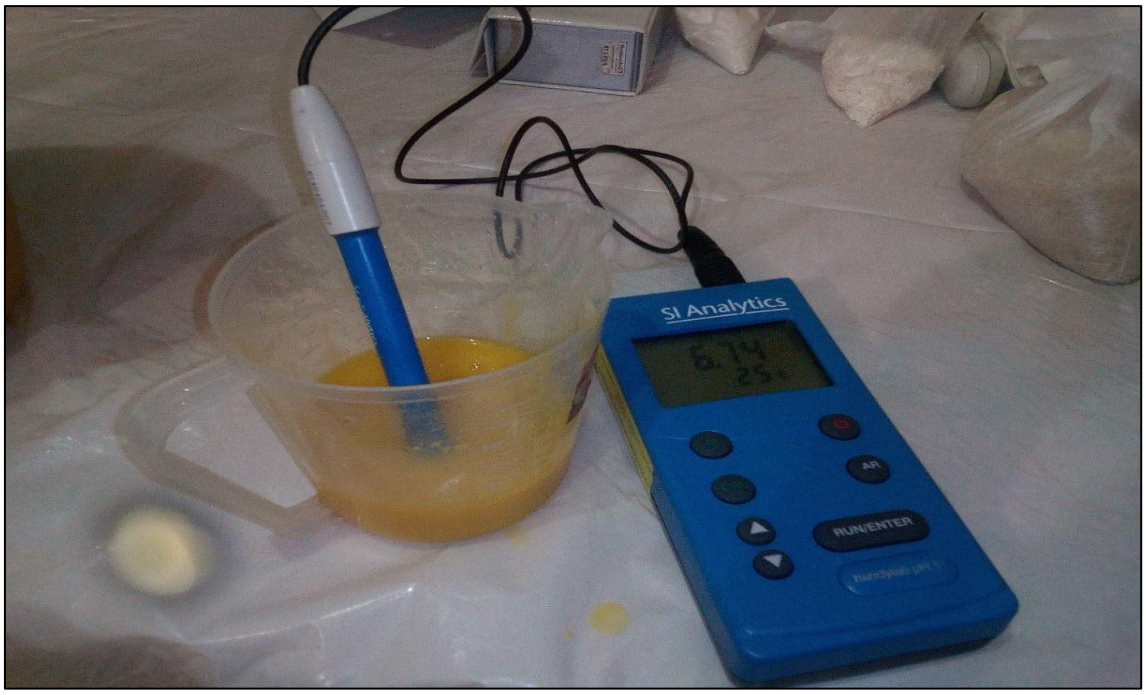

Figura 55. Análisis de pH.

Figura 57. Etiqueta.

Figura 58. Néctar de cocona.

Figura 59. Evaluación por alumnos.