Capítulo 1

De acuerdo al análisis es que se relaciona con tema 4 también se va relacionando con los temas o subtemas 5.3 y 6.3 ya que los temas se están relacionando con el método formales de la ingeniería de software zona de conocimiento de los modelos y métodos, en el análisis nos informa que hubo un gran impacto de inteligencia, ya que se quiere tener un lenguaje sea formalmente semántica definida, que con el análisis se requiere con un dos beneficios, el primero quiere la expresión de requisitos expresados en el idioma que se debe especificar de manera precisa, en segundo lugar deben ser razonado que permitan las propiedades deseadas del software y se debe especificar que se va utilizar y como se va probar y cuanto razonamiento se requiere que las herramientas sean factibles para cualquier cosa que sean sistemas triviales ya que se dividen en dos tipos que son probadores de teoremas o damas de modelo.

En ningún de los casos se puede realizar o plantear una prueba automatizada ya que se debe ver el razonamiento necesario de las herramientas, los análisis se centran por lo general en etapas tardías de análisis de requisitos, cuando se aplican la formalización de hasta objetos de negocios y los requisitos de los usuarios han llegado a un enfoque agudo a través de medios tales, ya que una vez que los requisitos se estabilizan han sido elaborados para que se especifique las propiedades concretas del software, que sea un beneficio formalización al menos los requisitos críticos ya que esto va permitir la validación estática que el software específica.

En este documento su definición del sistema es que los requisitos o conceptos de operaciones de documentos se van a registrar a los requisitos del sistema y esto hace una definición de los requisitos de alto nivel de la perspectiva del dominio y el marketing puede desempeñar estos roles para el software impulsados por el mercado.

Las herramientas de requisitos de software que tiene dos tipos de categorías: herramientas para moldear y herramientas para gestionar, las herramientas de gestión de requisitos suelen admitir una variedad de actividades que incluyen un documento y seguimiento la gestión de cambios que se van ir generando que tendrá un impacto de práctica, el seguimiento y la gestión de cambios solo son factibles si están respaldados por una herramienta ya que los requisitos, la gestión es fundamental para una buena práctica de requisitos muchas organizaciones han invertido en herramientas de gestión de requisitos y generan menos satisfacción tienen un ejemplo que es utilizando las hojas de cálculo.

Capítulo 2

Varias herramientas y técnicas que pueden ayudar a analizar y evaluar la calidad del diseño de software que tiene técnicas informales y formalizadas para determinar la calidad de artefactos por diseño un ejemplo seria la arquitectura revisiones, requisitos de rastreo, las revisiones de diseño de software también pueden evaluar la seguridad ayuda para las instalaciones, operaciones y uso por ejemplo los manuales y archivos de ayuda, y tiene la ayudo de evaluar la seguridad.

Análisis estático: está formado por formal o semiformal que tiene un análisis de diseño que se puede utilizar para la evaluación de un diseño el ejemplo seria el análisis de un árbol de fallas o una verificación cruzada automatizada, al hacer un diseño de análisis de vulnerabilidad se va ir formando un diseño de análisis de vulnerabilidad por ejemplo análisis estático de las debilidades de seguridad al realizarse la seguridad es una preocupación, al formar un análisis de diseño utiliza modelos matemáticos que permiten a los diseñadores predicar el comportamiento y validar el rendimiento del software y tener que dejar de depender de las pruebas.

Ya que en un análisis de pruebas se puede detectar especificación residual y errores de diseño que los puede generar la imprecisión, ambigüedad y a veces otros tipos de errores

Capítulo 3

La forma en que se está manejando los errores afecta al software capacidad para cumplir con los requisitos relacionados con la corrección, solidez y otro atributos no funcionales las afirmaciones a veces se utilizan para comprobar por errores, ya que otras técnicas de manejo de errores como devolviendo un valor neutral, sustituyendo el siguientes pieza de datos validos que van registrando un mensaje de advertencia que también se utiliza un código de error o cerrar el software.

Las excepciones se utilizan para detectar y procesar errores o eventos excepcionales, la estructura básica que tiene una excepción que es una rutina usa throw para lanzar un error detectada y un bloqueo de manejo de excepciones detectara la excepción en un try-catch bloquear.

El bloqueo try-catch puede procesar la condición errónea en la rutina o puede regresar con un control a la rutina de llamada, el manejo de excepciones las políticas deben ser diseñadas cuidadosamente siguiendo principios comunes como la inclusión en la que el mensaje de excepciones toda la información que condujo al excepción, evitando los bloques de captura vacíos, sabiendo las excepciones que arrojan el código de la biblioteca, tal vez creando un reporte de excepciones del programa.

Las estrategias de tolerancia a veces fallan las más comunes son incluir la realización de copias de seguridad y reintentar, utilizando auxiliares códigos, usando algoritmos de votación y reemplazando un valor erróneo con un valor falso que tendrá un efecto benigno.

Capítulo 4

La combinando funcional y estructural

Técnicas de prueba basadas en modelos y códigos a menudo se contrastan como funcionales versus estructurales pruebas estos dos enfoques para probar la selección no deben verse como alternativas sino más bien como complementos de hecho utilizando diferentes fuentes de información y se ha demostrado que destacan diferentes tipos de problemas en los pueden ver ser utilizados una combinación dependiendo del presupuesto consideraciones.

Determinista versus aleatorio

Los casos de prueba se deben seleccionar de forma determinista de acuerdo con una de las muchas técnicas o extraídas al azar de alguna distribución de entradas como suele hacer en las pruebas de fiabilidad varias comparaciones analísticas y empíricas han realizado para analizar las condiciones que hacen en un enfoque sea más eficaz que el otro.

Puntuación de mutación

En las pruebas de mutación se deben consultar las pruebas de mutación en la sección 3.4, técnicas basadas en fallas la relación de mutantes muertos al número total de generados.

Comparación y efectividad relativa

De diferentes técnicas se han realizado varios estudios para comparar la efectividad relativa de diferentes pruebas, Técnicas Es importante ser preciso en cuanto a propiedad contra la que se están aplicando las técnicas juzgado; cuál es, por ejemplo, el significado exacto dado al término "eficacia"? Las posibles interpretaciones incluyen el número de pruebas necesarias para encontrar el primer fallo, la relación del número de fallas encontradas a través de pruebas a todas las fallas encontradas durante y después de la prueba, y cuánto se mejoró la confiabilidad. Se han realizado comparaciones analíticas y empíricas entre diferentes técnicas, realizado de acuerdo con cada una de las nociones de eficacia especificada anteriormente.

Capítulo 5

Las actividades de planificación de mantenimiento

Una actividad importante para el mantenimiento de software es planificar y deben abordar los problemas asociados con una serie de perspectivas de planificación, que incluye:

- Planificación empresarial (nivel organizativo),
- Planificación de mantenimiento (nivel de transición),
- Lanzamiento / planificación de la versión (nivel de software), y
- Planificación individual de solicitudes de cambio de software (Nivel de solicitud).

A un nivel de solicitud individual, la planificación es llevada a cabo durante el análisis de impacto, el lanzamiento o versión ya que la actividad de planificación requiere que el mantenedor debe tener:

- Recopilar las fechas de disponibilidad del individuo peticiones,
- Estar de acuerdo con los usuarios sobre el contenido de los siguientes lanzamientos / versiones,
- Identificar conflictos potenciales y desarrollar alternativas,
- Evaluar el riesgo de una liberación determinada y desarrollar un plan de retirada en caso de que surjan problemas levantarse
- Informar a todas las partes interesadas.

Mientras los proyectos de desarrollo de software puedan normalmente duran desde algunos meses hasta algunos años, la fase de mantenimiento suele durar muchos años, hacer estimaciones de recursos es un elemento clave de la planificación del mantenimiento.

Capítulo 6

Tablero de control de configuración de software la autoridad para aceptar o rechazar propuestas los cambios recaen en una entidad conocida normalmente como, Tablero de control de configuración (CCB). En menor proyecto, esta autoridad en realidad puede residir con el líder o un individuo asignado en lugar de un tablero de varias personas. Puede haber varios niveles de autoridad de cambio dependiendo de una variedad de criterios, como la criticidad del elemento involucrado, la naturaleza del cambio (por ejemplo, impacto en presupuesto y cronograma), o el proyecto actual punto en el ciclo de vida. La composición de la los CCB utilizados para un sistema determinado varían según sobre estos criterios (un representante de SCM estar siempre presente). Todas las partes interesadas, apropiadas al nivel del CCB, están representados. Cuando el ámbito de autoridad de un CCB es estrictamente software, se conoce como configuración de software Tablero de control (SCCB). Las actividades del CCB suelen estar sujetos a auditorías de calidad de software o revisión.

Contabilidad del estado de la configuración del software (SCSA) es un elemento de la gestión de la configuración que consiste en el registro y la generación de informes de la información necesaria para gestionar una configuración de forma eficaz.

Información de estado de configuración del software

La actividad de la SCSA diseña y opera un sistema para la captura y reporte de información a medida que avanza el ciclo de vida. Como en cualquier sistema de información, se debe identificar, recopilar y mantener la información del estado de la configuración que se administrará para las configuraciones en evolución.

Se necesitan diversas informaciones y medidas para apoyar el proceso de SCM y para satisfacer las necesidades de gestión de informes de estado de configuración, ingeniería de software y otras actividades relacionadas.

Los tipos de información disponibles incluyen identificación de configuración aprobada, así como la identificación y el estado actual de implementación de cambios, desviaciones y exenciones.

Es necesaria alguna forma de soporte de herramientas automatizado para lograr la recopilación de datos de SCSA e informes de tareas; esto podría ser una capacidad de base de datos, una herramienta independiente o una capacidad de una entorno de herramientas integrado.

4.2. Informe de estado de configuración de software

La información reportada puede ser utilizada por varios elementos organizativos y del proyecto, incluidos el equipo de desarrollo, el equipo de mantenimiento, gestión de proyectos y actividades de calidad del software. Los informes pueden tomar la forma de consultas ad hoc para responder preguntas específicas.

Capítulo 7

Cierre

Un proyecto completo, una fase importante de un proyecto, o un ciclo de desarrollo iterativo llegan al cierre cuando todos los planes y procesos han sido promulgados y completado. Los criterios para el proyecto, debe evaluarse el éxito de la fase o iteración. Una vez que se establece el cierre, las actividades de archivo, retrospectiva y mejora de procesos pueden ser realizado.

Determinación del cierre

El cierre ocurre cuando las tareas especificadas para un se ha completado un proyecto, una fase o una iteración y se ha confirmado el cumplimiento satisfactorio de los criterios de finalización. Software los requisitos se pueden confirmar como satisfechos o no, y el grado de consecución de los objetivos puede ser determinado. Los procesos de cierre deben involucrar partes interesadas relevantes y dar como resultado la documentación de la aceptación de las partes interesadas relevantes; cualquiera conocido los problemas deben documentarse.

Actividades de cierre

Una vez confirmado el cierre, el archivo de los materiales del proyecto deben realizarse en de acuerdo con los métodos acordados por las partes interesadas, la ubicación y la duración, posiblemente incluyendo destrucción de información sensible, software, y el medio en el que residen las copias.

La base de datos de medición de la organización debe actualizarse con los datos relevantes del proyecto. Un proyecto, el análisis retrospectivo de fase o iteración debe emprenderse de modo que cuestiones, problemas, riesgos, y las oportunidades encontradas se pueden analizar (ver tema 4, Revisión y evaluación). Lección aprendida debe extraerse del proyecto y alimentarse en el aprendizaje y la mejora organizacional esfuerzos.

Planifique el proceso de medición

• Caracterizar la unidad organizativa. Los la unidad organizativa proporciona el contexto para medición, por lo que el contexto organizacional debe hacerse explícito, incluyendo las limitaciones que la organización impone a el proceso de medición. La caracterización se puede establecer en términos de organización procesos, dominios de aplicación, tecnología, interfaces organizacionales y organizacionales estructura.

• Identificar necesidades de información. Información las necesidades se basan en los objetivos, limitaciones, riesgos y problemas de la organización unidad. Pueden derivarse de negocios, organizacional, regulatorio y / o producto objetivos. Deben identificarse.

Capítulo 8

La medición de productos y procesos de software son preocupado por determinar la eficiencia y efectividad de un proceso, actividad o tarea. La eficiencia de un proceso de software, actividad, o tarea es la proporción de recursos realmente consumidos a los recursos que se espera o se desea consumir en la realización de un proceso, actividad o tarea (consulte Eficiencia en la ingeniería de software

Economía KA). El esfuerzo (o costo equivalente) es el medida principal de recursos para la mayoría del software procesos, actividades y tareas; se mide en

Unidades tales como horas-persona, días-persona, semanas-personal o meses-personal de esfuerzo o su equivalente unidades monetarias, como euros o dólares.

La eficacia es la relación entre la producción real y resultado esperado producido por un proceso de software, actividad o tarea; por ejemplo, el número real de defectos detectados y corregidos durante el software pruebas al número esperado de defectos a ser detectado y corregido, quizás basado en datos históricos para proyectos similares (ver Efectividad en la Ingeniería de Software Economía KA).

Tenga en cuenta que la medición de la eficacia del proceso de software requiere la medición de los atributos del producto; por ejemplo, medición de defectos de software descubiertos y corregidos durante pruebas de software.

Hay que tener cuidado al medir el producto.

Atributos con el propósito de determinar el proceso. Eficacia. Por ejemplo, el número de defectos detectado y corregido mediante pruebas puede no lograr el número esperado de defectos y así proporcionar una medida de eficacia engañosamente baja, ya sea porque el software que se está probando tiene una calidad mejor que la habitual o tal vez porque se ha introducido una inspección previa recién introducida proceso ha reducido el número restante de defectos en el software.

Medidas del producto que pueden ser importantes en determinar la efectividad de los procesos de software incluyen la complejidad del producto, defectos totales, densidad de defectos y calidad de los requisitos, documentación de diseño y otros trabajos relacionados productos.

La calidad del proceso y la medición del producto. los resultados se determinan principalmente por la confiabilidad y validez de los resultados medidos. Medidas que no cumplen estos criterios de calidad puede resultar en interpretaciones incorrectas y defectuosas iniciativas de mejora de procesos de software. Otra propiedad deseable de las mediciones de software incluir la facilidad de recopilación, análisis y presentación, además de una fuerte correlación entre la causa y efecto.

El tema de medición de ingeniería de software en el Software Engineering Management KA describe un proceso para implementar un software programa de medición.

Capítulo 9

Los métodos ágiles nacieron en la década de 1990 a partir de necesidad de reducir la aparente gran sobrecarga asociada con los métodos pesados basados en planes utilizados en proyectos de desarrollo de software a gran escala.

Los métodos ágiles se consideran métodos ligeros en el sentido de que se caracterizan por ciclos de desarrollo cortos e iterativos, equipos auto organizados, diseños más simples, refactorización de código, basado en pruebas desarrollo, participación frecuente del cliente y un énfasis en la creación de un trabajo demostrable producto con cada ciclo de desarrollo.

En la literatura se encuentran disponibles muchos métodos ágiles; algunos de los enfoques más populares, que se discuten aquí brevemente, incluyen Rapid

Desarrollo de aplicaciones (RAD), eXtreme Programming (XP), Scrum y Feature-Driven

Desarrollo (FDD).

• RAD: métodos rápidos de desarrollo de software se utilizan principalmente en el desarrollo de aplicaciones de sistemas empresariales con uso intensivo de datos. El RAD El método está habilitado con herramientas de desarrollo de bases de datos de propósito especial utilizadas por software ingenieros para desarrollar, probar e implementar rápidamente aplicaciones comerciales nuevas o modificadas.

• XP: este enfoque utiliza historias o escenarios para requisitos, primero desarrolla pruebas, tiene participación directa del cliente en el equipo (que normalmente define las pruebas de aceptación), utiliza programación por pares, y proporciona una refactorización e integración continuas del código. Cuentos se descomponen en tareas, se priorizan, estiman, desarrollan y prueban. Cada incremento de software se prueba con y pruebas manuales; un incremento puede ser lanzado con frecuencia, como cada par de semanas más o menos.

• Scrum: este enfoque ágil es más proyecto amigable con la administración que los demás. Los scrum master gestiona las actividades dentro el incremento del proyecto; cada incremento es llamado sprint y no dura más de 30 dias. Una lista de elementos de la cartera de productos (PBI) es desarrollado a partir del cual se identifican tareas definido, priorizado y estimado. Se prueba una versión funcional del software y liberado en cada incremento. Scrum diario

Las reuniones garantizan que el trabajo se gestione según la planificación.

• FDD: este es un enfoque de desarrollo de software iterativo, corto y basado en modelos que utiliza un proceso de cinco fases: (1) desarrollar un producto modelo para abarcar la amplitud del dominio, (2) crear la lista de necesidades o características, (3) construir el plan de desarrollo de funciones, (4) desarrollar

diseños para características específicas de iteración, y (5) codifique, pruebe y luego integre las funciones.

Capítulo 10

Requisitos de calidad del software

Factores de influencia

Varios factores influyen en la planificación, gestión, y selección de actividades y técnicas de SQM, incluso

- El dominio del sistema en el que reside el software; las funciones del sistema pueden ser seguridad crítica, misión crítica, negocio crítico, seguridad crítica
- El entorno físico en el que reside el sistema de software
- Sistema y software funcional (lo que el sistema) y la calidad (qué tan bien el sistema realiza sus funciones) requisitos
- Los componentes comerciales (externos) o estándar (internos) que se utilizarán en el sistema
- Los estándares específicos de ingeniería de software aplicable
- Los métodos y herramientas de software que se utilizarán para desarrollo y mantenimiento y para la evaluación y mejora de la calidad
- El presupuesto, el personal, la organización del proyecto, los planes, y programación de todos los procesos
- Los usuarios previstos y el uso del sistema
- El nivel de integridad del sistema.

La información sobre estos factores influye en cómo los procesos de SQM están organizados y documentados, cómo se seleccionan las actividades específicas de SQM, qué recursos se necesitan y cuáles de esos los recursos imponen límites a los esfuerzos

Herramientas de calidad de software

Las herramientas de calidad del software incluyen estáticas y dinámicas herramientas de análisis. Fuente de entrada de herramientas de análisis estático codificar, realizar análisis sintáctico y semántico sin ejecutar el código y presentar los resultados a usuarios. Existe una gran variedad en la profundidad, minuciosidad y alcance de las herramientas de análisis estático que

Calidad del software 10-13

Se puede aplicar a artefactos, incluidos modelos, en además del código fuente. (Consulte los KA de construcción de software, pruebas de software y mantenimiento de software para obtener descripciones de análisis dinámico herramientas.)

Las categorías de herramientas de análisis estático incluyen siguiendo:

• Herramientas que facilitan y automatizan parcialmente revisiones e inspecciones de documentos y código. Estas herramientas pueden en rutar el trabajo a diferentes participantes para automatizar parcialmente y controlar un proceso de revisión. Ellos permiten que los usuarios ingresen los defectos encontrados durante las inspecciones y revisiones para su posterior eliminación.

• Algunas herramientas ayudan a las organizaciones a realizar análisis de riesgos de seguridad del software. Estas herramientas proporcionar, por ejemplo, soporte automatizado para fallas análisis de modos y efectos (FMEA) y fallas análisis de árboles (FTA).

Capítulo 11

Dinámica de grupo y psicología

El trabajo de ingeniería se realiza muy a menudo en contexto de trabajo en equipo. Un ingeniero de software debe ser capaz de interactuar de manera cooperativa y constructiva con otros para determinar primero y luego satisfacer tanto las necesidades como las expectativas. Conocimiento de

La dinámica de grupo y la psicología es un activo cuando interactuar con clientes, compañeros de trabajo, proveedores, y subordinados para resolver la ingeniería de software problemas.

Dinámica de trabajo en equipos / grupos

Los ingenieros de software deben trabajar con otros. En por un lado, trabajan internamente en ingeniería equipos; Por otro lado, trabajan con clientes, miembros del público, reguladores y otras partes interesadas. Equipos de actuación: aquellos que demuestren una calidad constante de trabajo y progreso hacia las metas: son cohesivos y poseen un ambiente cooperativo, honesto y centrado.

Los objetivos individuales y de equipo están alineados para que los miembros se comprometen naturalmente y se sienten dueños de resultados compartidos.

Los miembros del equipo facilitan esta atmósfera al ser intelectualmente honesto, hacer uso del grupo pensar, admitir ignorancia y reconocer errores. Comparten responsabilidades, recompensas, y carga de trabajo de manera justa. Se encargan de comunicarse de forma clara, directa entre sí y en documentos, así como en código fuente, para que la información sea accesible para todos. Comentarios de pares sobre

Los productos de trabajo se enmarcan en un marco constructivo y forma no personal (consulte Revisiones y auditorías en el

Calidad del software KA). Esto permite que todos los miembros sigan un ciclo de mejora continua. Y crecimiento sin riesgo personal. En general, los miembros de equipos cohesionados demuestran respeto el uno para el otro y su líder.

Capítulo 12

Las estimaciones se utilizan para analizar y pronosticar los recursos o tiempo necesarios para implementar los requisitos (consulte Esfuerzo, cronograma y estimación de costos en el Software Engineering Management KA y estimación de costos de mantenimiento en el software

Mantenimiento KA). Cinco familias de estimación existen técnicas:

• Juicio experto

• Analogía

• Estimación por partes

• Métodos paramétricos

• Métodos de estadística.

Ninguna técnica de estimación es perfecta, por lo que es útil utilizar una técnica de estimación múltiple.

Convergencia entre las estimaciones producidas por diferentes técnicas indica que las estimaciones probablemente sean precisos. La dispersión entre las estimaciones indica que ciertos factores podrían haber sido pasado por alto. Encontrar los factores que causaron la propagación y luego volver a estimar para producir resultados que converjan podrían conducir a una mejor estimar.

Abordar la incertidumbre

Debido a los muchos factores desconocidos durante inicio y planificación del proyecto, las estimaciones son inherentemente incierto; que la incertidumbre debe ser abordados en las decisiones comerciales. Técnicas para abordar la incertidumbre incluyen:

• considerar rangos de estimaciones

• analizar la sensibilidad a los cambios de supuestos

• retrasar las decisiones finales.

La priorización implica clasificar alternativas basadas sobre criterios comunes para ofrecer lo mejor posible valor. En proyectos de ingeniería de software, software a menudo se priorizan los requisitos para entregar el mayor valor al cliente dentro de las limitaciones de cronograma, presupuesto, recursos y tecnología, o para proporcionar incrementos de productos de construcción, donde los primeros incrementos proporcionan el valor más alto para el cliente (ver Requisitos

Clasificación y negociación de requisitos en los requisitos de software KA y software Modelos de ciclo de vida en la ingeniería de software Proceso KA).

Las decisiones bajo técnicas de riesgo se utilizan cuando el tomador de decisiones puede asignar probabilidades a la diferentes resultados posibles (consulte Gestión de riesgos en la Gestión de ingeniería de software KA). Las técnicas específicas incluyen:

• toma de decisiones de valor esperado

• Variación de expectativas y toma de decisiones

• Análisis de Monte Carlo

• Árboles de decisión

• Valor esperado de la información perfecta.

Capítulo 13

Desde la perspectiva de una computadora, una amplia

Existe una brecha semántica entre su comportamiento previsto y el funcionamiento de la electrónica subyacente. Dispositivos que realmente hacen el trabajo dentro de la computadora. Esta brecha se cierra a través de la organización informática, que combina varios dispositivos eléctricos, electrónicos y mecánicos en un solo dispositivo. Que forma una computadora. Los objetos que la computadora la organización se ocupa de los dispositivos, conexiones y controles. La abstracción incorporada en la organización informática es la computadora.

Descripción general de la organización informática

Una computadora generalmente consta de una CPU, memoria, dispositivos de entrada y dispositivos de salida. Abstractamente hablando, la organización de una computadora puede ser dividida en cuatro niveles (Figura 13.4). La macro

El nivel de arquitectura es la especificación formal de todas las funciones que una máquina en particular puede realizar y se conoce como arquitectura de conjunto de instrucciones (ES UN). El nivel de micro arquitectura es la implementación de ISA en una CPU específica, en otras palabras, la forma en que las especificaciones de la ISA se llevan a cabo realmente. El nivel de los circuitos lógicos es el nivel donde cada componente funcional del micro arquitectura se compone de circuitos que toman decisiones basadas en reglas simples. El nivel de dispositivos es el nivel donde, finalmente, cada lógica El circuito está construido con dispositivos electrónicos como como semiconductores complementarios de óxido de metal (CMOS), semiconductores de óxido metálico de canal n (NMOS) o transistores de arseniuro de galio (GaAs).

Cada nivel proporciona una abstracción al nivel superior y depende del nivel inferior. A un programador, la abstracción más importante es la ISA, que especifica cosas como el nativo tipos de datos, instrucciones, registros, direccionamiento

modos, la arquitectura de memoria, interrupción y manejo de excepciones y las E / S. En general, el ISA especifica la capacidad de una computadora y qué se puede hacer en la computadora con programación.

En el nivel más bajo, se realizan cálculos por los dispositivos eléctricos y electrónicos dentro de un computadora. La computadora usa circuitos y memoria para contener cargas que representan la presencia o ausencia de voltaje. La presencia de voltaje es igual a 1 mientras que la ausencia de voltaje es un cero. En el disco, la polaridad del voltaje está representada por 0 y 1 que a su vez representa los datos almacenados. Todo, incluida la instrucción y datos: se expresan o codifican mediante ceros digitales y unos. En este sentido, una computadora se convierte en un sistema digital. Por ejemplo, el valor decimal 6 puede codificarse como 110, la instrucción de suma puede codificarse como 0001, y así sucesivamente. El componente de la computadora como la unidad de control, ALU, memoria y E / S utilizan la información para calcular las instrucciones.

Capítulo 14

La gramática de un lenguaje natural nos dice si una combinación de palabras hace una válida frase. A diferencia de los lenguajes naturales, un lenguaje formal se especifica mediante un conjunto bien definido de reglas para sintaxis. Las oraciones válidas de un lenguaje formal pueden ser descritas por una gramática con la ayuda de estas reglas, conocidas como reglas de producción.

Un lenguaje formal es un conjunto de longitud finita palabras o cadenas sobre un alfabeto finito, y una gramática especifica las reglas para la formación de estas palabras o cadenas. El conjunto completo de palabras que son válidos para una gramática constituye el lenguaje de la gramática. Por tanto, la gramática G es cualquier definición matemática compacta y precisa de un lenguaje L en lugar de solo una lista sin procesar de todos de las oraciones legales del idioma o ejemplos de esas oraciones.

Una gramática implica un algoritmo que

Generar todas las oraciones legales del idioma.

Existen diferentes tipos de gramáticas.

Una estructura de frase o gramática de tipo 0 G = (V, T, S, P) es una tupla de 4

El término libre de contexto deriva del hecho de que

A siempre se puede reemplazar por a, independientemente de la contexto en el que ocurre.

Un lenguaje formal está libre de contexto si lo genera una gramática libre de contexto. Los lenguajes libres de contexto son la base teórica de la sintaxis de la mayoría de los lenguajes de programación. Gramática regular. Todos los

fragmentos en el RHS son terminales simples o un par construido por un terminal y no terminal; es decir, si A → a, entonces ya sea a ∈ T, o a = cD, o a = Dc para c ∈ T, D ∈ N.

Si a = cD, entonces la gramática se llama derecha gramática lineal. Por otro lado, si a = Dc, entonces la gramática se llama gramática lineal izquierda. Ambos las gramáticas lineal derecha e izquierda son gramática regular o tipo 3.

El lenguaje L (G) generado por un la gramática G se llama lenguaje regular.

Una expresión regular A es una cadena (o patrón) formado a partir de las siguientes seis piezas de información: a ∈ S, el conjunto de alfabetos, e, 0 y el operaciones, O (+), PRODUCTO (.), CONCATENACIÓN (*). El idioma de G, L (G) es igual a todas esas cadenas que coinciden con G, L (G) = {x ∈ S * | x coincide con G}.

Para cualquier a ∈ S, L (a) = a; L (e) = {ε}; L (0) = 0. + funciona como un o, L (A + B) = L (A) ∪ L (B). . Crea una estructura de producto, L (AB) = L (A).L (B).

* denota concatenación, L (A *) = {x1x2… Xn |

xi ∈ L (A) y n ³ 0}

Por ejemplo, la expresión regular (ab) *coincide con el conjunto de cadenas: {e, ab, abab, abababab,ababababab,…}.