PROGRAMACION DE ROBOTS Y LENGUAJES

INTRODUCCIÓN

En las máquinas controladas por sistemas informáticos, el lenguaje es el medio que utiliza el hombrepara gobernar su funcionamiento, por lo que su correcta adaptación con la tarea a realizar y la sencillez de manejo, son factores determinantes del rendimiento obtenido en los robots industriales.

Hay varias maneras de comunicarse con un robot, y tres soluciones generales para lograrlo, que son *reconocimiento de palabras separadas, enseñanza y repetición y lenguajes de programación de alto nivel.*

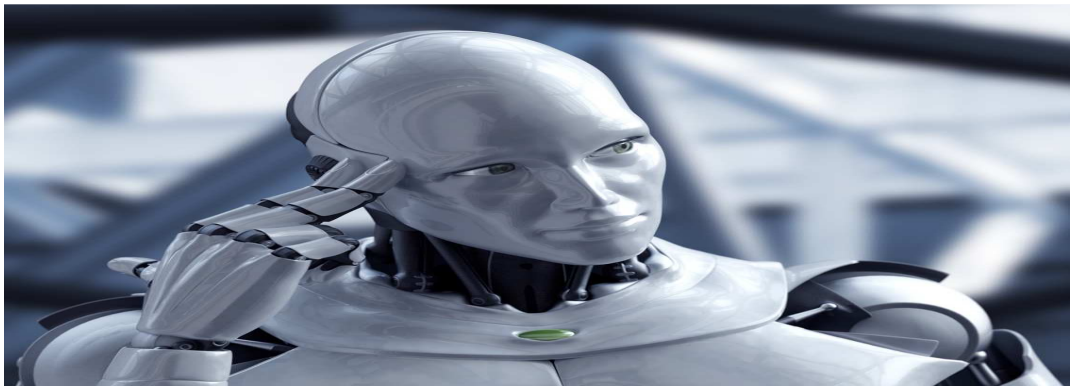
Los sistemas de reconocimiento de la voz en la tecnología moderna son bastante primitivos y suelen depender de quien habla. Estos sist emas pueden reconocer un conjunto de palabras concretas de un vocabulario muy limitado y en general exigen al usuario una pausa entre las palabras, aunque en la actualidad es posible reconocer las palabras separadas en tiempo real debido a los cada vez más rápidos componentes de las computadoras y algoritmos de procesamiento más eficientes, la utilidad del reconocimiento de palabras separadas para describir la tarea de un robot es bastante limitada.

La enseñanza y repetición, también conocido como g uiado, es la solución más común utilizada en el presente para los robots industriales.

Guiar al robot en movimiento lento, puede ser en general llevado a cabo de varias maneras: usando un joystick, un conjunto de botones (uno para cada movimiento) o un sistema de manipulación maestro-esclavo.

Los lenguajes de programación de alto nivel suministran una solución más general para resolver el problema de comunicación hombre-robot. Los lenguajes clásicos empleados en informática, como el FORTRAN, BASIC, PASCAL, etc., no disponen de las instrucciones y comandos específicos que necesitan los robots, para aproximarse a su configuración y a los trabajos que han de realizar..

La estructura del sistema informático del robot varía not ablemente, según el nivel y complejidad del lenguaje y de la base de datos que requiera.



....el objetivo más importante es.... !Comprender y Construir Entidades Inteligentes!

Programación del robot software y Lenguajes

Programación

Es un conjunto de reglas para comunicar ideas , generalmente las ideas se le comunican a una máquina.

PROGRAMACION DEL ROBOT

Un robot hoy en día puede hacer mucho más que mover su brazo a lo largo de una serie de puntos dentro de un espacio. Los robots de tecnología actual pueden aceptar datos de entrada procedentes de sensores y otros dispositivos. Pueden enviar señales a elementos del equipo que operan con ellos dentro de la célula. Pueden tomar decisiones. Pueden comunicarse con otras computadoras para recibir instrucciones y para informar sobre los datos de producción y los problemas. Todas estas capacidades necesitan de la p rogramación.

El gran desafío es Hombre y máquina interactúen juntos ya que necesitan uno del otro para solucionar eficazmente los problemas

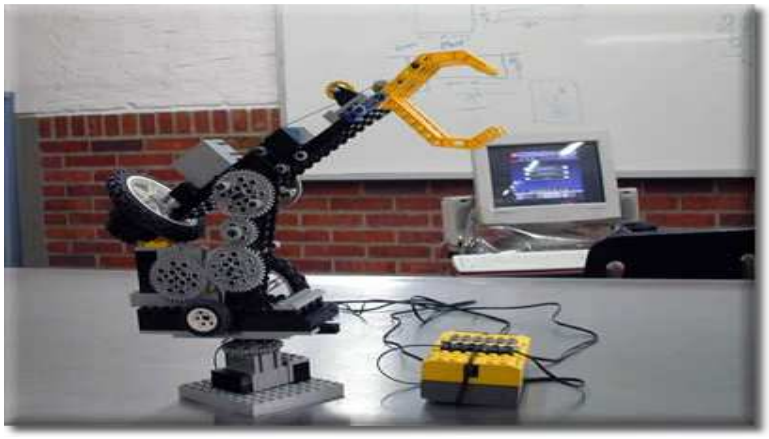
METODOS DE PROGRAMACION DEL ROBOT

La programación del robot se realiza de varias formas. En coherencia con la práctica industrial actual, dividiremos los métodos de programación en dos tipos básicos:

1. Métodos de aprendizaje directo.
2. Lenguajes textuales del robot.

Los métodos de aprend izaje directo requieren que el programador mueva el manipulador a lo largo del camino del movimiento que se desea almacenar en la memoria por medio del controlador del robot. Los métodos de aprendizaje directo hacen referencia, algunas veces, a los métodos de «enseñar mostrando». Cronológicamente, los métodos de aprendizaje directo representan los primeros métodos reales de programación de robots utilizados en la industria. Tienen sus

comienzos en los primeros años 60 cuando se utilizaron los primeros robot s para aplicaciones industriales. La programación de robot con lenguajes textuales se realiza, en cierto modo, lo mismo que la programación de computadora. El programador introduce por teclado el programa en un monitor TRC (tubo de rayos catódicos) utilizando un lenguaje de alto nivel similar al inglés. Este procedimiento se suele completar con la util ización de técnicas de aprendi zaje directo para señalar al robot la posición de los puntos dentro del espacio de trabajo. Los lenguajes textuales comenzaron a desarrollarse en la década de los 70 con la primera aparición de un lenguaje comercial hacia el año 1979. Además de los lenguajes de programación de aprendizaje directo *y* textuales, se utiliza otro método para simplificar la programación de los robots de baja tecnología. Nos referimos a estos tipos de máquinas como robots de secuencia limitada, que se controlan por medio de topes mecánicos e interruptores de fin de carrera para definir los puntos extremos de los movimientos de sus articulaciones. Al posici onamiento de estos topes mecánicos e interruptores se le podría denominar método de programación.



Utilización del lenguaje de programación del robot Karel, el cual es un subconjunto de Pascal; este es utilizado por la introducción a la enseñanza de la programación.

Métodos de definición de posiciones en un espacio

Prescindiendo de la configuración del robot, existen varios métodos que el programador puede utilizar durante el modo de aprendizaje para mover el brazo del robot y la muñeca. He aquí una lista con los tres métodos siguientes:

1. Movimientos de articulación.

2. Movimientos de coordenadas *x-y-z* (también denominadas coordenadas universales).

3. Movimiento de coordenadas de herramientas.

El primero es el método más básico y suele referirse al movimiento de cada una de las articulaciones, por medio de un dispositivo suspendido de enseñanza (control de mandos). Este dispositivo tiene un conjunto de interruptores basculantes (o dispositivos de control similar) para hacer funcionar a cada una de las articulaciones en una u otra de sus dos direcciones hasta que el efector lineal haya alcanzado la posición deseada. Este método de enseñanza de puntos se refiere, con frecuencia, como el modo de la articulación. Los sucesivos posicionamientos del brazo del robot

para definir una secuencia de puntos pueden ser consumidora de tiempo, resultando tedioso programar el robot. Para superar esta desventaja, algunos robots se pueden controlar durante el modo de aprendizaje para desplazarse dentro de los movimientos de coordenadas *x-y-z.* Este método, denominado sistema de coordenadas universales, permite definir la localización de la muñeca utilizando un

sistema convencional de coordenadas cartesianas con orige n en alguna posición en el cuerpo del robot. En el caso del robot de coordenadas cartesianas, este método equivale prácticamente al modo de la articulación de programación. Para los robots polares, cilíndricos y de brazo articulado, el controlador debe res olver un conjunto de ecuaciones matemáticas para convertir los movimientos de la articulación rotacional del robot en el sistema de coordenadas cartesianas. Estas conversiones son realizadas de tal modo que el programador no tiene que preocuparse por los c álculos importantes, que están siendo desarrollados por el controlador. Para el programador, la muñeca (o el efector final) se está desplazando en movi - mientos que son paralelos a los ejes x, *y y z.* Las dos o tres articulaciones adicionales, que constituy en la unión de la muñeca, son casi siempre rotacionales y mientras que la programación se hace en el sistema *x-y-z* para mover el brazo y las articulaciones del cuerpo, la muñeca suele ser mantenida por el controlador en una orientación constante.

ilustra el método *x-y-z* de definir los puntos en el espacio para un robot de brazo articulado.

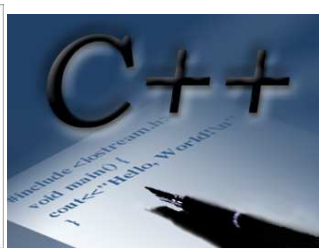
Algunos robots tienen la capacidad de definir los movimientos de coordenadas de herramienta. Este es un sistema de coordenadas cartesianas en el que el origen está localizado en algún punto dentro de la muñeca y el plano *xy* es paralelo a la placa frontal de la muñeca. En consecuencia, el eje *z* es perpendicular a la placa frontal y está apuntando en la misma dirección que la herramienta u otro efector final unido a la placa frontal. Por tanto, este método de movimiento del robot podría utilizarse para proporcionar un movimiento impulsor de la he rramienta. De nuevo, el controlador debe realizar una cantidad significativa de esfuerzo de cálculo para permitir al programador utilizar los movimientos de la muñeca para definir movimientos puntos.

La Figura muestra el sistema de coordenadas de la herramienta.

• De que hay que estar pendiente cuando programamos en un lenguaje:

- Sintaxis / Semantica
- Sistema de tipos
- Errores / Excepciones

• Lenguajes de programación populares:

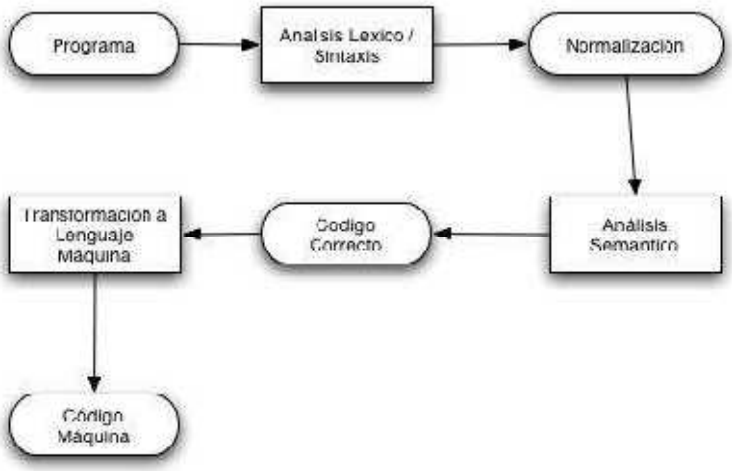


- C, C++, Java, PHP, Perl, XHTML.

• Dominios de aplicación importantes:
Programación Sistema
Sistemas de Gestión de Información
Programación Web
Niveles de los lenguajes

Lenguaje Natural
Lenguaje de Programación
Compilador / Interprete
Lenguaje Máquina

Etapas de la Compilación

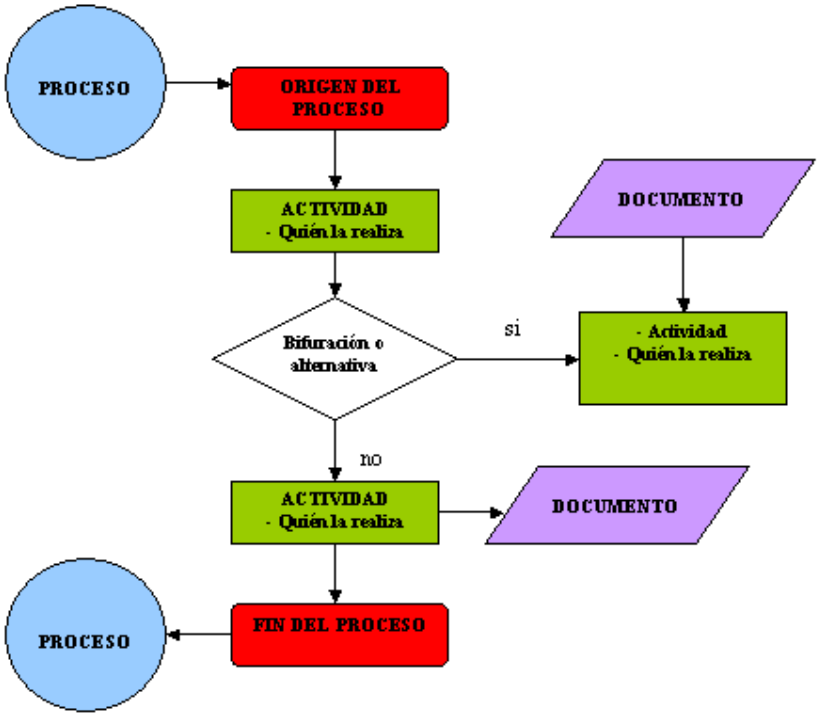


Lenguaje de Máquina

• Lenguaje mas básico, propio de cada computadora,ya que está relacionado con el diseño del hardware de la misma (dependiente de la máquina). Por lo general consisten en cadenas de números al final reducidos a ceros y unos (sistema numérico binario).

• Operaciones:

• Cargar
• Almacenar
• Sumar
• Restar



Grafica: Diagramas de Flujo, N-S.- Un Diagrama de flujo es la representación detallada en forma gráfica de un algoritmo

Lenguaje Ensamblador

• Consiste en abreviaturas similares al inglés, llamadas instrucciones mnemotécnicas, que permiten representar las operaciones elementales de la computadora (dependiente de la máquina).

Ejemplo:

Código de operación	Dirección ensamblador	Instrucción en lenguaje
00010101	10000001	LOAD A
00010111	10000010	ADD B
00010110	10000011	STORE

• *Lenguaje de bajo nivel o ensamblador* :

La computadora no entiende directamente lenguaje ensamblador por lo que un programa escrito en este lenguaje tiene que ser traducido a lenguaje de máquina por un programa llamado un *ensamblador* para que pueda ser ejecutado por la computadora.

Los lenguajes ensambladores todavía requieren que el programador tenga un buen conocimiento de la arquitectura de la computadora. Como los lenguajes ensambladores son dependientes de la máquina, todo programa es crito en un

lenguaje ensamblador particular tendrá que ser reescrito si se va a ejecutar en otro tipo de computadora.

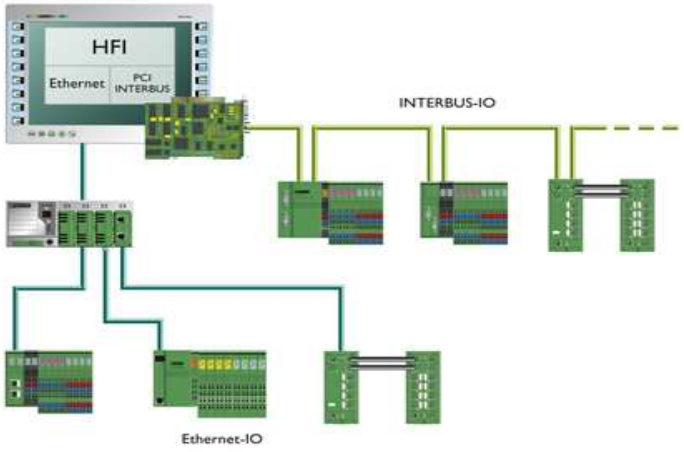
Lenguaje de Alto Nivel

• Permite a los programadores escribir instrucciones en un lenguaje mas familiar para ellos y que contiene notaciones matemáticas comúnmente utilizadas (independiente de la máquina).

Ejemplo:

Código de en operación	Dirección	Instrucción en lenguaje ensamblador	Instrucción lenguaje de alto nivel
00010101	10000001	LOAD A	
00010111	10000010	ADD B	
00010110	10000011	STORE C	C = A + B

Con este tipo de lenguajes, la programación es mas fácil para los usuarios ya que éste no necesita tener conocimiento de la arquitectura de la computadora.



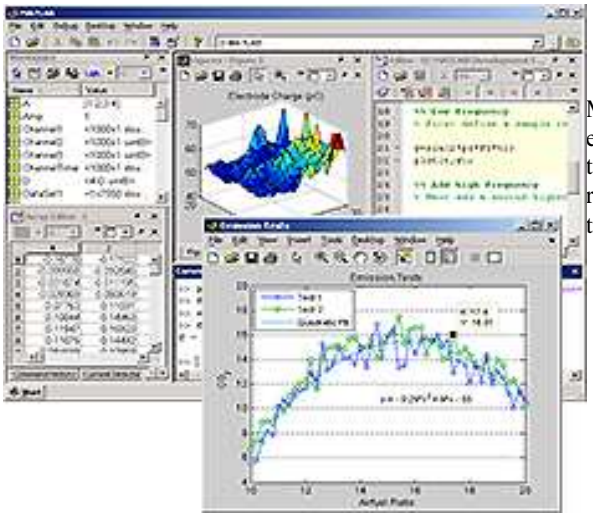
Lenguaje de alto nivel Sistemas de mando de lenguaje de alto nivel con unidad de operación integrada (CP HLC)

Como ocurre con los lenguajes ensambladores, la computadora no entiende directamente lenguaje de alto nivel, por lo que un programa escrito en este lenguaje tiene que ser traducido a lenguaje de máquina por un programa llamado un *compilador* para que pueda ser ejecutado por la computadora.

Los lenguajes de alto nivel permiten portabilidad, mejor expresión de las ideas, facilidad de programar ciertas clases de problemas, menos posibilidad de cometer errores, una visión más amplia del problema, etc.

Ejemplos de lenguajes de alto nivel :
- Java - BASIC
- C - Visual Basic

- C++ - Pascal
- COBOL
- FORTRAN
- PROLOG
- LISP



MATLAB es un lenguaje de alto nivel y un entorno interactivo que le permite realizar tareas de cálculo complejas de forma más rápida que con los lenguajes de programación tradicionales, como C, C++ y Fortran

Cada CPU tiene su propio lenguaje de máquina interno. La programación a este nivel se realiza generalmente en el lenguaje ensamblador específico de la computadora. Cada instrucción en lenguaje ensamblador corresponde a una instrucción en lenguaje de máquina. Si existe una estandarización para un lenguaje de alto nivel, cualquier programa escrito usando este estándar debe poder ejecutarse en cualquier computadora después de compilarlo. Esto se le conoce como *portabilidad* de programas.

Elementos de un lenguaje de programación

Un sub lenguaje para definir los datos

- Qué datos tenemos
- Cómo les llamamos
- Cómo son (tipo y/o estructura)
- Qué se puede hacer con ellos

Un sub lenguaje para definir los algoritmos

- Qué le hacemos a los datos
- En qué orden (cuándo se lo hacemos)
- Cuántas veces

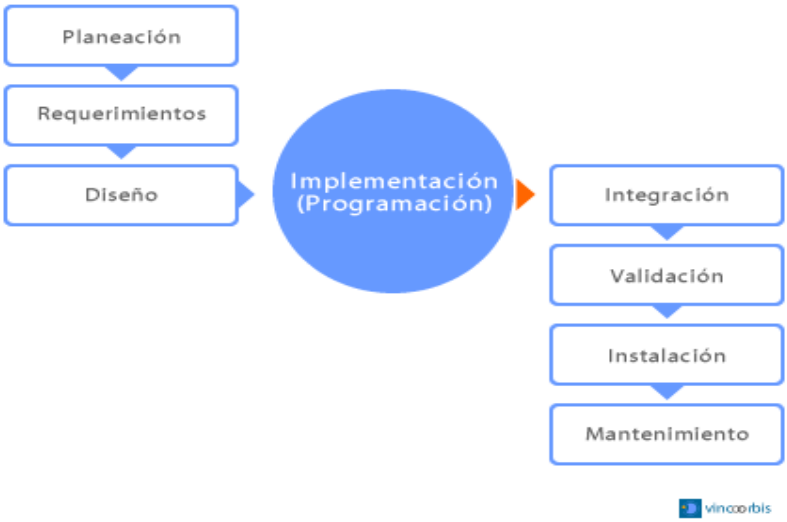
Metodología de Desarrollo de Programas

El desarrollo de programas sigue hoy en dia distintas metodologías: De arriba hacia abajo, espiral, modular, etc.

Diseño del algoritmo: Descripción de una secuencia finita y ordenada de pasos – sin ambigüedades – que conducen a la solución de un problema dado.

Herramientas de diseño

- Diagramas de flujo (para la programación estructurada)
- Círculos y canales de mensaje (programación orientada a objetos)
- Pseudocódigo
- Trazas personales
- Grafismos
- Formulas matemáticas
- Todo aquello que le ayude a repr esentar el problema.



Desarrollo de Software y Programación de Aplicaciones

Codificación: Traducción del algoritmo a un programa escrito en un lenguaje de programación adecuado (código fuente).

Corrida en frío del programa : Prueba manual de la correctitud del programa.

Depuración del programa : Identificación y eliminación de errores.

- Errores de sintaxis: Violan las reglas del lenguajede programación. Un buen compilador localizará e identificará la mayoría de estos automáticamente.
- Errores lógicos: Equivocaciones que causan que el programa se ejecute de forma inesperada o incorrecta.

- *Ejecución del programa*: Ejecución del códigoejecutable (código en lenguaje de máquina) del programa bajo el control del CPU, una instrucción a la vez.
- *Puesta en operación*: Instalación del hardware y software, capacitación, etc..
- *Mantenimiento del programa* : Comienza tan pronto como el producto es lanzado. Permite corregir defectos menores, añadir una mayor funcionalidad, ya sea en respuesta a las demandas del mercado o a las peticiones del usuario.

Análisis E-P-S

– *Especificaciones de entrada:* Información necesaria para la solución del problema.
- ¿ Qué datos son de entrada ?
- ¿ Cuántos datos se introducirán ?
- ¿ Cuáles datos de entrada son válidos ?

– *Proceso:* Operaciones o cálculos necesarios para encontrar la solución del problema.
- ¿ Qué tipo de ecuaciones ?
- ¿ Cuántas ecuaciones ?
- ¿ Qué transformaciones sobre la data?

– *Especificaciones de salida:* Resultados finales de los cálculos.
- ¿ Cuáles son los datos de salida
- ¿ Cuántos datos de salida se producirán
- ¿ Qué precisión tendrán los resultados
- ¿ Se debe imprimir un encabezado

Diseño del algoritmo

Un algoritmo debe ser *preciso* e indicar el orden de realización de cada paso.

Un algoritmo debe ser *finito*. Si se sigue un algoritmo, se debe terminar en algún momento; o sea, debe tener un número finito de pasos.

Codificación

Traducir el algoritmo producido en el paso anterior en un programa escrito en un lenguaje de programación de alto nivel (programa fuente o có digo fuente). En nuestro caso C++.

Visual C++ 2008 Express Edition es un entorno de desarrollo para la creación de aplicaciones con C++, Visual C++ 2008 Express Edition te permite realizar el diseño y la codificación de sus aplicaciones creadas con el lenguaje C++ bajo la plataforma de Windows.

Los diferentes pasos de un algoritmo se expresan en los programas como instrucciones (término usado para los lenguajes de máquina y bajo nivel), sentencias oproposiciones (términos usados para los lenguajes de alto nivel).

Programa: Secuencia de sentencias, cada una de las cuales especifica ciertas operaciones que debe ejecutar la computadora.
Tipos básicos de sentencias:

- Sentencias de entrada/salida
- Sentencias aritmético-lógicas
- Sentencias de decisión o selectivas
- Sentencias repetitivas o lazos

Tipos básicos de sentencias

Sentencias de entrada/salida : Sentencias de transferencia de información y datos entre dispositivos de E/S (teclado, impresora, discos, etc.) y la memoria principal.

Sentencias aritmético-lógicas: Sentencias que ejecutan operaciones aritméticas (suma, resta, multiplicación, etc.) o lógicas (y lógico, o lógico, negación).

Sentencias de decisión o selectivas : Sentencias que permiten la selección de tareas alternativas en función de los resultados de diferentes expresiones condicionales.

Sentencias repetitivas o lazos : Sentencias que permiten la repetición de secuencias de sentencias un número determinado o indeterminado de veces.

Corrida en frío del programa

El programador realiza una corrida en frío sobre el programa fuente escogiendo un conjunto de datos de entrada, ejecutando manualmente cada sentencia del programa fuente y verificando que los resultados obtenidos son los esperados de acuerdo al conjunto de datos de entrada. Como una técnica de depuración, el programador debe realizar este proceso utilizando conjuntos de datos que permitan ejecutar todos los “caminos” posibles del programa.

Ejecución del programa

El programa fuente es introducido a la computadora utilizando un programa llamado *editor*.

Una vez editado, el programa fuente es traducido por el *compilador* a un programa escrito en lenguaje de máquina (código objeto), siempre y cuando el programa no tenga *errores de sintaxis* (errores de gramática).

Si el programa fuente tiene errores de sintaxis no se genera código objeto. Los errores deben ser corregidos usando el editor y luego se el programa fuente se debe volver a compilar.

Cuando el programa está sintácticamente correcto, el código objeto es encadenado con las funciones de librería (otros programas) requeridas usando un programa llamado *encadenador*.

El código objeto compilado y encadenado es cargado en la memoria principal para su ejecución por un programa llamado *cargador*.

El código objeto compilado, enlazado y cargado (código ejecutable) es *ejecutado* con los datos de entrada.

Comprobación del programa

Comprobar que el programa realiza las tareas para las cuales ha sido diseñado y produce el resultado correcto y esperado. Si el programa tiene *errores de lógica* (errores en el método de solución por lo que las salidas obtenidas no corresponden con las salidas esperadas), éstos deben ser corregidos en el programa fuente usando el editor, el cual se debe volver a compilar.

Inteligencia Artificial

Como toda disciplina de reciente creación, la IA no se encuentra unificada en términos de objetivos y métodos de investigación. Recientemente, parte de los esfuerzos de los investigadores en esta área se han dedicado a la definición de dichos o bjetivos y al recuento de. las herramientas metodológicas utilizadas hasta ahora (Boden, 1977; Dennett, 1978; Sloman, 1978; Ringle, 1979). Como resultado de este esfuerzo, que dista mucho de su conclusión, se han definido algunos acuerdos básicos sobre el área y sus estrategias.

Por ahora, es suficientemente claro que el objetivo de la IA es el de entender la naturaleza de la inteligencia a través del diseño de sistemas computacionales que la exhiban. En forma más concreta, puede afirmarse que, en lo que h a transcurrido de su corta historia, la IA ha estado dirigida por tres objetivos generales:

1. El análisis teórico de las posibles explicaciones del comportamiento inteligente

2. La explicación de habilidades mentales humanas

3. La construcción de artefactos (computadoras) inteligentes

Características de la IA

- Uso de símbolos no matemáticos, aunque no es suficiente para distinguirlo completamente.
- El comportamiento de los programas no es descrito explícitamente por el algoritmo.
- El razonamiento basado en el conocimiento, implica que estos programas incorporan factores y relaciones del mundo real y del ámbito del conocimiento en que ellos operan
- Aplicabilidad a datos y problemas mal estructurados

Objetivos de la IA

- Desarrollar una máquina intelig ente capaz de aprender a través de la experiencia

- Hacer que las computadoras sean capaces de mostrar un comportamiento que sea considerado como inteligente por parte de un observador humano
- Elevar el Coeficiente Intelectual de las máquinas (machine -IQ)
- Desarrollar las capacidades de la computadora más allá de su uso tradicional actual

Ramas de la IA

Con estos propósitos en su agenda de investigación, los estudiosos de la IA han recurrido al uso de cuatro diferentes estrategias metodológicas: el desarrollo de tecnologías útiles en esta área, la simulación, el modelamiento, y la construcción de teoría sobre la inteligencia artificial. El desarrollo de tecnologías de computación ha sido una empresa titánica que los ingenieros en electrónica han toma do en sus manos, sin embargo, sólo una pequeña parte de lo que se conoce como ciencia de la computación puede incluirse dentro de la IA. No existe todavía un criterio preciso con el cual distinguir cuándo un sistema computacional es un sistema de IA, pero el acuerdo general es que cualquier máquina que desempeñe una función mental que tendría que ser realizada por una inteligencia humana es un ejemplo de IA.

La simulación que se hace en IA ha intentado reproducir algunas de las características inteligentes de los seres humanos. Estas reproducciones han buscado abiertamente la similitud entre una computadora y los seres humanos. La elaboración de simulaciones ha sugerido la posibilidad de explorar los procesos cognoscitivos humanos, sin embargo los esfuerzos en esta línea, a diferencia del modelamiento, han estado dedicados a producir comportamiento humano inteligente en las computadoras más que a entenderlo o explicarlo.

El modelamiento, por otra parte, tiene como objeto la utilización de los sistemas de IA para entender a la inteligencia humana. Ha sido tradicionalmente utilizado por psicólogos y no tiene como requisito necesario el uso de computadoras, De hecho, muchas de las teorías sobre cognición han utilizado modelos en computadoras sin hacer referencia a ellas, por ejemplo, la teoría sobre memoria semántica o sobre representación mental.

Finalmente, el trabajo teórico en IA ha abierto por primera vez la posibilidad de teorizar sobre la inteligencia sin hacer necesariamente referencia a la inteligencia humana. Es decir, se ha propuesto la formulación de una teoría de la inteligencia "pura".

Una breve incursión dentro de la historia de esta disciplina ilustrará más adecuadamente sus logros y su estado actual.

Historia de la Inteligencia Artificial

Los esfuerzos por reproducir algunas habilidades mentales humanas en máquinas y androides se remontan muy atrás en la historia. El mito del coloso de Rodas entre los griegos, las estatuas "parlantes" del medioevo, el androide de Von Kempelen que jugó al ajedrez con Napoleón, y el "motor analítico" de Charles Babbage que calculaba logaritmos, son sólo algunos de los ejemplos de este antiguo interés. Igualmente, la concepción de la inteligencia humana como un mecanismo no es reciente ni ha estado disociada de la psicología: Descartes, Hobbes, Leibniz, y el mismo Hume se refirieron a la mente humana como una forma de mecanismo.

Durante el siglo XIX y la primera mitad del XX, las analogías biológicas y fenomenológicas desplazaron a la noción de mecanismo en el est udio de la mente humana. Sin embargo, a partir de la segunda mitad de nuestro siglo, la noción de mecanismo renovó su poder heurístico con la formalización de la noción de "computación".

Como algunas máquinas, especialmente las calculadoras, se diseñaron para evitar el tener que pensar y para hacer el pensamiento más rápido y exacto, fue inevitable que desde sus orígenes las calculadoras, y más adelante las computadoras, se relacionaran con la inteligencia y el pensamiento enfatizando sus similitudes.

La IA fue introducida a la comunidad científica en 1950 por el inglés Alan Turing en su artículo "Maquinaria Computacional e Inteligencia." A pesar de que la investigación sobre el diseño y las capacidades de las computadoras comenzaron algún tiempo antes, fue hasta que apareció el artículo de Turing que la idea de una máquina inteligente cautivó la atención de los científicos.

La pregunta básica que Turing trató de responder afirmativamente en su artículo era: ¿pueden las máquinas pensar? Los argumentos de T uring en favor de la posibilidad de inteligencia en las máquinas, iniciaron un intenso debate que marcó claramente la primera etapa de interacción entre la IA y la psicología. Los debates en aquella época se centraron en el análisis de la serie de problema s implicados en la aplicación de términos mentalistas a las computadoras. La intención de Turing no era la de usar estos términos como analogías sino la de eliminar la distinción entre inteligencia natural e inteligencia artificial.

Dos de las contribuciones más importantes de Turing a la IA fueron el diseño de la primera computadora capaz de jugar ajedrez y, más importante que esto, el establecimiento de la naturaleza simbólica de la computación.

El trabajo de Turing, quien falleció prematuramente, fue c ontinuado en los Estados Unidos por John Von Neumann durante la década de los cincuentas. Su contribución central fue la idea de que las computadoras deberían diseñarse tomando como modelo al cerebro humano. Von Neumann fue el primero en "antropomorfizar" el lenguaje y la concepción de la computación al hablar de la "memoria", los "sensores", etc., de las computadoras. Construyó una serie de máquinas utilizando lo que a principios de los cincuentas se conocía sobre el cerebro humano, y diseñó los primeros programas almacenados en la memoria de una computadora.

Alan Turing (1912 – 1954)

Sin embargo, esta línea de investigación pronto encontró serias limitaciones. La concentración en la imitación de la constitución físico -química del cerebro, no permitió ver, a Von Neumann y sus seguidores, que la analogía sería mucho más eficiente si se estudiaran las funciones del cerebro, es decir, sus capacidades como procesador de información.

Corresponde a McCulloch, a mediados de los cincuentas, formular una posici ón radicalmente distinta al sostener que las leyes que gobiernan al pensamiento deben buscarse entre las reglas que gobiernan a la información y no entre las que gobiernan a la materia. Esta idea abrió grandes posibilidades a la IA. En esta línea, Minsky (1959), uno de los padres fundadores de la IA, modificó su posición y sostuvo que la imitación del cerebro a nivel celular debería ser abandonada.

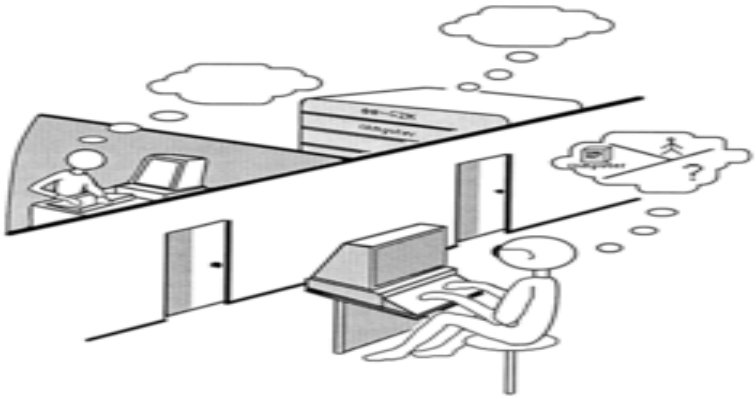
Es más o menos en esta época que ocurre un evento que organizaría y daría un gran impulso al desarrollo de la IA: el congreso en Darthmouth (1956). En este congreso, en el que se reunieron los padres fundadores de la disciplina, se llegó a la definición de las presuposiciones básicas del núcleo teórico de la IA:

1. El reconocimiento de que el pensamiento puede o currir fuera del cerebro, es decir, en máquinas

2. La presuposición de que el pensamiento puede ser comprendido de manera formal y científica

3. La presuposición de que la mejor forma de entenderlo es a través de computadoras digitales

Desde fines de los cincuentas la investigación en IA se expande y se multiplica en direcciones diversas. La capacidad simbólica de las computadoras es estudiada, entre otros, por Shanon (1950) y por Newell, Shaw y Simon (1958) quienes diseñan el primer programa inteligente basado en su modelo de procesamiento de información. Este modelo de Newell, Shaw y Simon habría de convertirse pronto en la teoría dominante en psicología cognoscitiva.



Prueba de Turing

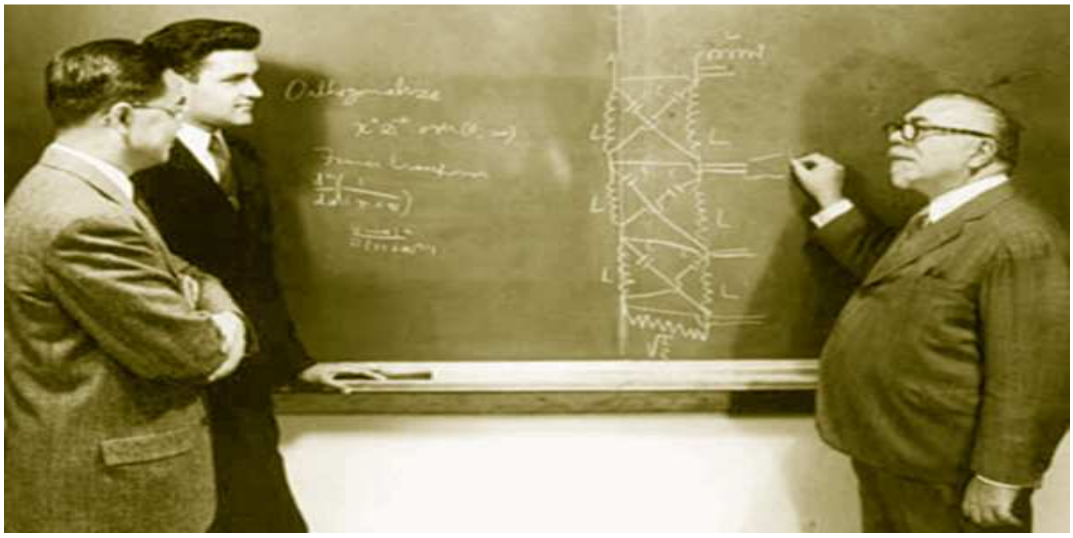
Algunos investigadores se dedicaron al estudio de la naturaleza del aprendizaje en las computadoras y a los procesos de reconocimiento de patrones visuales. Como resultado de ello Selfridge y Dinneen consiguen diseñar el primer programa capaz de aprender por experiencia (ver McCorduck, 1979).

Basándose en los estudios sobre memoria asociativa, el equipo Newell-Shaw-Simon construyó los primeros lenguajes de procesamiento de información (IPL-I, IPL-II) utilizados en el diseño de su "Logic Theorist Machine" que se convirtió en la primera máquina "inteligente". Esta máquina fue capaz no sólo de memorizar y aprender, sitio que consiguió demostrar de una manera original y "creativa", es decir no prevista por sus creadores, algunos de los teoremas propuestos por Bertrand Russell en los Principios (Russell and Whitehead, 1925).

Desde sus orígenes la IA se relacionó con juegos como el ajedrez y las damas, probablemente debido a que los juegos de mesa constituyen modelos de situaciones reales en las que hay que calcular, solucionar problemas, tomar decisiones, corregir errores, recordar, etc. A pesar de que esta línea de investigación ha sido casi totalmente abandonada en la actualidad, muchos de los avances teóricos y metodológicos de la IA se deben a ella. Por ejemplo, Samuel diseñó en 1961 un programa que jugaba damas y que era capaz de aprender de sus errores, es decir, era capaz de adaptar su comportamiento en relación a eventos pasados. Lo pasmoso de este programa fue que, aunada a su capacidad de aprendizaje la de memoria, con el tiempo consiguió derrotar invariablemente a su creador. El mismo resultado fue obtenido por Bernstein a través de un programa que jugaba ajedrez (Boden, 1977). Los grandes "retos" entre computadoras y seres humanos se multiplicaron, siendo el más famoso de ellos el que ocurrió entre Dreyfus (un encona do crítico de la IA) y el programa Machack, en el que Dreyfus fue derrotado en un juego de ajedrez de varias horas.

A principios de los sesentas, la IA comienza una fase distinta de su desarrollo. En 1962, McCarthy y Raphael inician sus trabajos sobre el diseño y la construcción de un robot móvil que llamarían "Shakey". La diferencia fundamental entre este robot y los programas en computadora utilizados hasta ahora por la IA, es que "Shakey" tendría que enfrentar el reto de interactuar con el mundo real en términos de espacio, tiempo, movimiento, etc. En otras palabras, el robot tendría que tener alguna forma de "conocimiento" del mundo que lo rodeaba. Este reto inició una fuerte preocupación en

la IA por el estudio de la epistemología y los procesos cogno scitivos. La discusión se centró alrededor de los problemas de la representación mental o interna del conocimiento, la percepción y los problemas del significado. La idea básica de Raphael era la de reunir, en una sola, distintas máquinas con capacidad de aprender por experiencia, de reconocer patrones visuales, de modelar, de manipular símbolos, etc., y esperar que el todo fuera mayor que la suma de las partes. El resultado del experimento no fue el éxito que Raphael esperaba, pero fue un logro sin precede nte que hizo posibles avances muy importantes. El aprendizaje más importante de esta experiencia fue la comprensión de que el problema más difícil por resolver en IA era el de construir una máquina capaz de funcionar con altos ni<,eles de incertidumbre, como lo hace un ser humano. Se hizo claro que construir una máquina que no lidiara efectivamente con la incertidumbre sería una de dos: o trivial, por la simpleza de la tarea, o imposible por la complejidad de la misma.



Hacia mediados de los sesentas la IA se convierte en un área en la que se interesan e interactúan especialistas de diversas disciplinas: lógicos, psicólogos, matemáticos, lingüistas, filósofos, etc.

Uno de los grandes temas de IA en esta década fue el estudio del lenguaje. En la mayoría de los estudios iniciales sobre lenguaje, se atacó el problema de diseñar una máquina que fuera capaz de traducir de un idioma a otro. El énfasis se hizo en el análisis de la sintaxis, en lugar del significado, estrategia que se abandonó relativamente pronto. Los investigadores interesados en esta área de la IA pronto descubrieron que traducción no es lo mismo que transformación, y que, como consecuencia de ello, de alguna manera la máquina tendría que "entender" un texto antes de poder traducirlo. Los esfuerz os se orientaron hacia una definición de la comprensión que pudiera simularse en una computadora.

Con esta idea como lineamiento heurístico, se desarrolló una serie de programas que exploraban esta serie de habilidades relacionadas con el lenguaje y la comprensión: el "oráculo" de Phillips, "Sad Sam" de Lindzay que fue uno de los más exitosos, y "Eliza" de Wizenbaum y Colby. (ver Boden, 1977). El programa de Lindzay se orientó hacia el estudio del significado, connotativo y denotativo, y es capaz de hace r inferencias, conocer, y parafrasear sobre el mundo conocido. Eliza, por otra parte, consistió en un análisis de la conversación que se basa en la simulación de una terapia psicológica interactiva en la que el programa es el analista y el interlocutor es el analizado. Un ejemplo de una conversación sostenida con Eliza en la computadora de la Universidad de Stanford es el siguiente (el diálogo de la computadora en letras mayúsculas):

BUENAS TARDES, POR FAVOR ESCRIBA SU NOMBRE

buenas tardes, me llamo Javie r

ES UN PLACER CONOCERTE, JAVIER

el placer es mio

¿QUE TE TRAE POR AQUI?

nada en especial, he tenido trabajo y me siento agotado

¿QUE TE HACE PENSAR QUE ESTAS AGOTADO?

he tenido problemas en mi trabajo

¿PROBLEMAS?

es decir, he estado preocupado con mi trabajo

CUENTAME TUS PREOCUPACIONES...

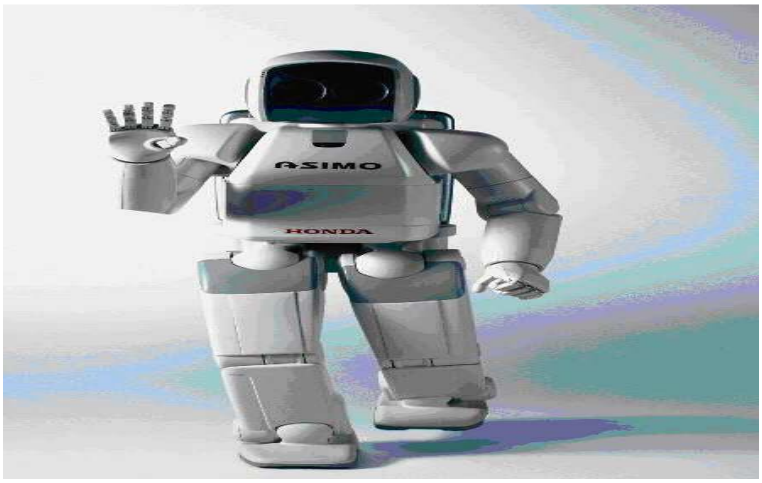
Según Colby, psicoanalista por entrenamiento, el programa resultó tan exitoso en términos clínicos que valdría la pena intentar reproducirlo masivamente como instrumento terapéu tico. Intento al que Weizenbaum, coautor del programa, se opuso decididamente con el argumento de que no puede decirse desde ningún punto de vista que "Eliza" es capaz de entender lo que se le dice. "Eliza", Weizenbaum insiste, es un programa capaz de resp onder, que no es lo mismo que comprender.

Como resultado de este y otros debates, las preguntas de investigación siguieron girando alrededor de la noción de comprensión. Los seres humanos comprendemos en una plática mucho más de lo que se nos dice explíci ta y verbalmente (en parte porque la comunicación no~verbal juega un papel muy importante y en parte por nuestras capacidades para obtener información del contexto y para operar con altos niveles de incertidumbre) y pareció importante que estas habilidades fueran reproducidas en una computadora con pretensiones de inteligencia. Los investigadores se dedicaron entonces al estudio de los aspectos no -verbales del lenguaje y al de reconocimiento de patrones.

Para mediados de los sesentas, y como resultado de l os estudios sobre lenguaje y comprensión. la IA se involucré en el estudio del problema de la representación mental del conocimiento. Igualmente, una gran preocupación en esta década fue la de encontrar aplicaciones prácticas de lo que se había aprendido h asta entonces. Entre las aplicaciones más exitosas debe mencionarse el diseño de DENDRAL en Stanford, un programa que hace espectogramas y diagnósticos médicos y que hasta ahora ha tenido la misma tasa de error que los seres humanos. La segunda aplicación importante tiene que ver con la psicología educativa y la mencionaré en la siguiente sección sobre la interacción entre la IA y la psicología.

Existe todavía una enorme cantidad de fenómenos psicológicos sin descripciones ni explicaciones adecuadas. Entre ellos: percepción, memoria, reconocimiento de patrones, comprensión, aprendizaje, comunicación, interpretación, creación, intuición, selección, etc. Los psicólogos, como la mayoría de los científicos sociales, han tenido hasta ahora un conjunto de herramientas inadecuadas para la formulación de teoría y la corroboración de hipótesis sobre esta serie de fenómenos. Algunos de ellos han tratado de recurrir sin gran éxito a modelos más recientes como la teoría de sistemas o la teoría de juegos que, por su naturaleza estática, siguen siendo deficientes en la explicación de la actividad mental.

La IA ha explorado las distintas formas en que las computadoras podrían realizar las tareas que antes estaban reservadas a los seres humanos, como resolver problemas, planear a futuro, demostrar teoremas, jugar ajedrez, conversar en y entender un lenguaje, componer música, cte. El hecho de que aún no haya conseguido reproducir un ser humano completo (o que eventualmente lo consiga) es de menor importancia que la evidencia de que ha mejorado nuestras habilidades para pensar y clarificar fenómenos de interés para la psicología y otras ramas de la ciencia. Ha colaborado en la reformulación de viejos problemas psicológicos y en la reconsideración y revaluación de las teorías existentes. Ha forzado a los especialistas a precisar sus conceptos y a utilizarlos con mayor consistencia y rigor, al mismo tiempo, los ha enfrentado a un reto teórico al formular un programa rival de investigación que busca explicar un objeto de estudio que anteriormente estaba reservado a los psicólogos: la explicación de la inteligencia y sus procesos colaterales.



La Universidad de Córdoba acogerá en junio de 2010 una importante reunión que acogerá a expertos internacionales sobre inteligencia artificial; se trata de la '*23rd Internacional Conferencia on Industrial, Engineering and Other Aplicattions os Applied Intelligent Systems'*.

Sus logros y avances han sido tan importantes que, en unos cuantos años, las teorías sobre la inteligencia y sus fenómenos relacionados serán literalmente incomprensibles para el psicólogo que no esté familiarizado con el desarrollo teórico, los métodos y los logros de la IA. Su influencia en la psicología ha sido y será tan grande que enseñar cursos universitarios sobre psicología cognoscitiva sin hacer referencia a la IA será un grave acto de ignorancia o de irresponsabilidad.

Conclusiones

Es mucho lo que se ha hecho en el área de la programación para la robótica; sin embargo aún no existe un lenguaje ideal para la programación de los robots. Son muchos los lenguajes creados hasta ahora, en parte las causas principales de esta amplia gama de lenguajes inadecuados o poco efectivos son:
Cada lenguaje se ha diseñado tomando como base un robot en específico del mercado, lo que anula su universalidad y la posibilidad de emplearlo en modelos diferentes.
Los lenguajes, en muchos casos, se dirigen hacia aplicaciones diferentes, lo que limita grandemente su utilización para la programación de otras tareas.
Hoy en el mundo existe un interés general para lograr un sistema de percepción del entorno cada vez más avanzado. Para esto se hace necesaria la ampliación de la Inteligencia Artificial, que interviene en la valoración del espacio exterior o entorno y determina los planes de acción alternativos o lo que es lo mismo la respuesta a la interacción con ese medio.

Características de un lenguaje ideal para la robótica
Las seis características básicas de un lenguaje ideal, expuestas por Pratt, son:
1. Claridad y sencillez.
2. Claridad de la estructura del programa.
3. Sencillez de aplicación.
4. Facilidad de ampliación.
5. Facilidad de corrección y mantenimiento.
6. Eficacia.

Estas características son insuficientes para la creación de un lenguaje "universal" de programación en la robótica, por lo que es preciso añadir las siguientes:
- Transportabilidad sobre cualquier equipo mecánico o informático.
- Adaptabilidad a sensores (tacto, visión, etc.).
- Posibilidad de descripción de todo tipo de herramientas acoplables al manipulador.
- Interacción con otros sistemas.

Bibliografía :

Evolución artificial y robótica autónoma, José Santos.Richard J.Duro editorial:Ra-Ma primera edicion.

Introducción a la automática y mecánica de robots, E.Sanchez Muños.editorial.Servicio de publicaciones Universidad de Cádiz. segunda edición.

http://www.slideshare.net/marias099/sesion1-historia-definiciones-panorama-actual

http://www.slideshare.net/Alumnos/robotica-167676?src=related_normal&rel=866336

http://eleconomista.com.mx/notas-online/tecnociencia/2009/01/21/robots-cada-vez-mas-importantes-salas-cirugia

http://www.google.com/imgres?imgurl=http://www.monografias.com/trabajos31/robotica/ro3.jpg&imgrefurl=http://www.monografias.com/trabajos31/robotica/robotica.shtml&u