Marcos Metodológicos dentro de la Informática

Pablo Iuliano[1], Luis Marrone[1], and Elvio Fernandez

LINTI, Facultad de Informática UNLP,
La Plata, Argentina
{piuliano,lmarrone}@info.unlp.edu.ar

Abstract. El presente documento abordará e indentificará dos corrientes filosóficas contrapuestas dentro de la ciencia, el Racionalismo y el Empirísmo, en relación con la actividad informática. En primera instancia se describirán las corrientes filosóficas antes citadas, para luego realizar un análisis en cual se vinculará las corrientes filosóficas a casos concretos dentro de la disciplina en cuestión. Finalmente se presentarán las conclusiones obtenidas.

Keywords: Racionalismo; Empírismo; Informática; Métodos;

1 Marco histrico-filosfico

En este apartado se introducirán las tendencias filosóficas de interés para este trabajo. Lejos de ser un estudio comparativo de distintas ramas de la filosofía de la ciencia se tocarán los aspectos más representativos y necesarios para su posterior vinculación con la visión y la actividad de la informática.

1.1 El siglo XVII

Paralelamente con las leyes de Johannes Kepler sobre el movimiento de los planetas y sus órbitas y el descubrimiento por Galileo Galilei de la ley de caída libre de los cuerpos se desarrollan los escritos de Francis Bacon, en los que enfatiza su idea de *gran renovación* de la ciencia a partir de la definición de un *nuevo instrumento* que aparta los prejuicios de los espíritus de los hombres. Esta crítica de Bacon, a la vieja ciencia, es la precursora de dos grandes vertientes de la *nueva filosofía* del siglo XVII. El racionalismo cartesiano y la interpretación empirista de la naturaleza.

1.2 Racionalismo

"En el siglo XVII la comunidad de posturas, en lo que respecta a la posibilidad del conocimiento y a su esencia, no se da en la cuestión del origen del conocimiento. El racionalismo coloca a la razón en su origen al aseverar que el pensamiento es la fuente y el fundamento del conocimiento. Ello es así porque responde a las exigencias de necesidad lógica y de validez universal..." [1].

Para argumentar esta *validez universal* el racionalismo plantea que la razón proporciona juicios que pueden ser demostrables. Aquellas expresiones de las que se puede decir que son evidentemente y necesariamente ciertas o validas o, por el contrario, que son forzosamente falsas o invalidas, las cuales encierran una verdad necesaria. Mientras que las experiencias (emociones y sensaciones) proporcionan juicios que expresan únicamente la posibilidad/ambigüedad de algo, y de las cuales pueden pensarse lo contrario. De esta manera el racionalismo enuncia que el verdadero conocimiento y la naturaleza de la realidad tiene un único y necesario origen: el racional.
El innatismo postulado por Descartes, enuncia que el entendimiento es fundado en una ilustración del espíritu por Dios. Existen ideas innatas a partir de las cuales los hombres establecen relaciones lógicas entre proposiciones que constituyen parte de la actividad de la razón.
"... He notado ciertas leyes que Dios ha establecido en la naturaleza y cuyas nociones ha impreso en nuestras almas, de tal suerte que, si reflexionamos sobre ellas. Con bastante detenimiento, no podremos dudar de que se cumplen exactamente en todo lo que es o se hace en el mundo" [2].
La *idea innata* es solo una de las formas de lo que Descartes llama sustancia. La sustancia existe de tal manera que no tiene necesidad de otra cosa para existir [2]. Se afirma la existencia de tres sustancias:

1. La sustancia finita pensante (el ser pensante).
2. La sustancia extensa (mundo)
3. El propio Dios, sustancia infinita pensante.

Son tres sustancias pero sólo dos modos de ser sustancia: el pensamiento y la extensión o materia. Resulta relevante el término de *sustancia* si lo traducimos con el concepto de idea. Especialmente la noción de la existencia de la idea de *"yo como sujeto pensante"*. A diferencia de la sustancia extensa, como la idea del mundo como objeto del conocimiento; en cuanto a Dios, es garantía que nuestro conocimiento está en perfecta concordancia con el orden de la realidad. Es Dios quien pone en forma de ideas innatas los pensamientos claves para que el hombre alcance el conocimiento a través de la razón.
En la física y las demás ciencias basadas en el estudio de la realidad extensa, lo mensurable y analizable según conceptos geométricos, Descartes aplico el modelo teórico aportando la hipótesis de que el mundo inanimado y los cuerpos animados son máquinas (mecanicismo), y que todos los fenómenos físicos son explicables fundamentados en este principio. El fuerte carácter de la filosofía mecanicista cartesiana fascinó a los estudiosos de la época, incluso a los empiristas [1].
La influencia del mecanicismo continúa hasta la primera revolución industrial -situado desde el siglo XVII al XIX- teniendo como símbolo los engranajes [3]. Puramente mecanicista, la organización es vista como una máquina solamente para satisfacer los deseos de lucro. Sólo posteriormente en la segunda revolución se puede observar el desarrollo de los procesos industriales devenidos de *engranajes* a *cadena de montajes*.

1.3 Empirismo

En contraposición, con el pensamiento cartesiano, el empirísmo rechaza (antiinnatismo) de raíz todo iluminismo, de toda fuerza trascendente que pueda postularse como ilustrativa del alma humana.
Para Thomas Hobbes no existe el dualismo mente-cuerpo que enuncia el pensamiento cartesiano. Esta dualidad es resuelta mediante la reducción de ambos en un único principio material. Las características propias de la mente no escapan a las leyes que rigen las demás cosas. Entonces, la fuente para conocer los objetos materiales está en las sensaciones suministradas por nuestros sentidos, los cuales constituyen el único criterio de verdad. Así, la razón opera únicamente con nombres, que no son otra cosa que pura convención, y el lenguaje es el instrumento necesario para la adquisición del conocimiento. Ello sucede recordando los hechos de la experiencia, operando sobre ellos según ciertas reglas y comunicándolos a otros individuos. Razón y lenguaje, desde este punto de vista, aparecen tan artificiales o convencionales como lo pueda ser la sociedad.
La no admisión de conceptos universales, ni en las cosas ni en la mente, conduce a la doctrina sensualista de Hobbes a definir toda actividad intelectiva como un puro movimiento. Entendiendo este, como la actividad más primordial/básica en la mente, aquello que dispara la imaginación siendo pensado como un movimiento en los órganos del cuerpo.
John Locke, añade que las ideas simples proceden directamente de la experiencia, para alimentar así la *tabla rasa* de la mente. Y las ideas complejas se componen de las anteriores mediante su combinación, yuxtaposición o abstracción. Estas operaciones mentales se llevan a cabo según leyes aportadas por la experiencia; la razón es guía de todo conocimiento probable y no tiene otro límite que la experiencia. De esta manera Locke señala que el conocimiento posible solo puede suceder a *posteriori*. El conocimiento es basado en la experiencia.
A diferencia del empirismo, el racionalismo no ve limite en la razón. Ésta, con los métodos adecuados, permitiría el abordaje y entendimiento del *todo*. El empirismo, por el contrario, sitúa al hombre como alguien acotado forzando un corrimiento en el foco desde el hombre hacia el fenomeno/contexto a observar. La experiencia es el límite en el entendimiento de la complejidad de la realidad.

2 Racionalismo dentro de la Informática

Como se verá a continuación, dentro de la informática aún son notables las influencias de las ideas y abordajes del siglo XVII. El mecanicismo y racionalismo de Descartes están entrelazados de manera indivisible en la historia de la informática y por consiguiente en los métodos y técnicas del quehacer informático.

2.1 Desarrollo en Cascada

Una de las técnicas más utilizadas a lo largo de la historia del desarrollo de software es la conocida como desarrollo en cascada o modelo en cascada. Este

es un enfoque metodológico que ordena rigurosamente las etapas del proceso de desarrollo de software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior. El desarrollo en cascada esta compuesto por las siguientes fases o etapas:

1. Análisis de requisitos: En esta etapa se analizan los requerimientos que deberá cumplir el software mediante una o varias entrevistas con el o los usuarios a los cuales esta dirigido el software y así determinar qué objetivos debe cumplir el mismo. De esta fase surge un documento con la especificación completa de lo que debe hacer el sistema sin entrar en detalles de implementación. Cabe señalar que en esta etapa se consensua todo lo que se requiere del sistema y no pudiéndose solicitar nuevos requisitos a mitad del proceso de elaboración del software.
2. Diseño del sistema: Aqui se descompone y organiza el sistema en elementos que puedan elaborarse por separado, aprovechando las ventajas del desarrollo en equipo. Como resultado surge un documento, que contiene la descripción de la estructura relacional global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la manera en que se combinan unas con otras.
3. Codificación: Es la fase en donde se implementa el código fuente, haciendo uso de un lenguaje de programación. Dependiendo del lenguaje utilizado y su versión se crean las bibliotecas y componentes reutilizables dentro del mismo proyecto para hacer que la programación sea un proceso mucho más rápido.
4. Pruebas y Verificación: Los elementos, ya programados, se ensamblan para componer el sistema, se comprueba que funciona correctamente y que cumple con los requisitos, antes de ser entregado al usuario final.
5. Mantenimiento: Es la etapa más extensa de todo el proceso de desarrollo. El sistema se instala y se pone en funcionamiento corrigiendo todos los errores no descubiertos en las etapas anteriores y se realizan mejoras de implementación. Llegado el caso se identifican nuevos requisitos.

Reminiscencias racionalistas en el desarrollo en cascada En proyectos de desarrollo de software de la vida real, rara vez se sigue una secuencia lineal, esto crea un deficiente analisis de requisitos lo cual es el punta pie inicial para obtener una mala implementación del sistema y concluyendo en una etapa de mantenimiento practicamente insostenible, lo cual generalmente conduce al fracaso.
Al ser un proceso lineal-secuencial que no permite la reformulación de etapas ya transitadas, lo diseñado en las etapas tempranas del proyecto es lo que será finalmente el sistema consumado.
Existe una sola dirección en la formulación del análisis/diseño. No existe ninguna etapa de contrastación con la realidad, asumiendo la actividad del desarrollo como un proceso inyectivo análisis/diseño $\Rightarrow$ codificación/mantenimiento.
Puede notarse en este tipo de abordaje las reminiscencias del pensamiento racionalista donde la razón era la única herramienta valida para las construcciones int-

electuales. Este modelo de desarrollo de software posiciona las etapas asociadas al raciocinio del sistema en las etapas más tempranas del proceso y relegando la acción de construcción para el final del mismo. Así se deja entre ver que el desarrollo en cascada esta inspirado en la frase *"Pienso luego Existo"* o en este caso *"Pienso luego desarrollo"*. Otra consideración relevante es que la etapa del mantenimiento se posiciona en el final del proceso de desarrollo, la cual termina siendo una etapa de adecuación de lo ideado a la realidad.

3 Empírismo dentro de la Informática

Las aproximaciones empiristas dentro de la informática muchas veces están relacionadas con los abordajes de la complejidad y los sistemas abiertos [4]. Lo estudiado no es fácilmente asimilable debido a la naturaleza no lineal del objeto de estudio.
Esta última afirmación queda debidamente evidenciada a partir de los casos presentados a continuación.

3.1 Desarrollo de Aplicaciones de Simulación

Las aplicaciones orientadas a intentar resolver problemas complejos, o al menos proveer un medio para poder entenderlos, son en general implementadas como herramientas computacionales de simulación.
Pero antes de describir las posibles opciones para implementar herramientas de este tipo se debe definir que se entiende por simulación, para dicho fin a continuación se citará una definición perteneciente a Shannon Robert para el concepto antes mencionado:
"La simulación es el proceso de diseñar y desarrollar un modelo computarizado de un sistema o proceso y conducir experimentos con este modelo con el propósito de entender el comportamiento del sistema o evaluar varias estrategias con las cuales se puede operar el sistema"[5].
De la definición anterior surgen dos conceptos, modelo y proceso:

1. Modelo de simulación: conjunto de hipótesis acerca del funcionamiento del sistema expresado como relaciones matemáticas y/o lógicas entre los elementos del sistema.
2. Proceso de simulación: ejecución del modelo a través del tiempo en una computadora para generar muestras representativas del comportamiento.

La simulación como método empírico Claramente las aplicaciones de simulación tienen una naturaleza netamente empírica ya que no se conoce de antemano los resultados a los cuales se podrán arribar antes de su ejecución. Así como la construcción de un simulador también implica transitar un proceso sistematico de prueba y error que se reitera tantas veces hasta que la aplicación se ajuste lo suficientemente al problema del mundo real que se intenta responder o entender según sea el caso.

3.2 Transición de IP versión 4 a IP versión 6

Las mayoria de las redes pequeñas, medianas y grandes funcionan utilizando un protocolo de red llamado IP (*Internet Protocol*). Este protocolo de red es en el cual se basa la transmisión de datos en Internet y su definición se encuentra especificada en la RFC 971.[7]
Para que los equipos, por ejemplo computadoras e impresoras, que forman parte de una red IP se puedan comunicar, cada uno de ellos debe tener asignado una dirección IP que los identifique univocamente y así poder establecer comunicación con otros equipos.
La versión que se usa actualmente para el direccionamiento IP es IPv4. Esta versión cuenta con direcciones de 32 bits de longitud, con lo cual se obtiene un total de 4.294.967.296 direcciones disponibles. Dentro de las cuales, hay un gran número de ellas destinadas para usos reservados y no se pueden utilizar para ser asignadas a dispositivos de red.

El problema con IPv4 El surgimiento de nuevas tecnologías y de nuevos dispositivos que requieren conectividad a Internet, han hecho que la demanda de direcciones IP crezca de forma grotesca. Smartphones, camaras con acceso a internet, etc. han hecho que la vida útil de esta versión se reduzca de forma drástica cada vez más. En un principio se pronosticó que las direcciones IP se acabarían cerca del año 1995, pero gracias a diversas técnicas se ha pospuesto dicha proyección para finales del año 2013 aproximadamente.

La solución al agotamiento de direcciones IP Teniendo en mente la problematica del agotamiento de las direcciones IP y después de examinar diferentes propuestas, la Internet Engineering Task Force (IETF) decidó adoptar la propuesta recomendada en 1995 y definida en el RFC 1752.[8] Esta propuesta aparece como una nueva versión del protocolo IP y debido a que se realizaron varias pruebas, extensiones y modificaciones a la versión 4 con el fin de resolver el problema en cuestión, para evitar confusiones futuras a la nueva versión se le asignó el número 6. De esta manera surgió la próxima generación de IP denominada IPv6. La caracteristica más relevante de IPv6 es que provee direcciones de 128 bits de longitud y con esto desaparece por completo el problema de agotamiento de direcciones ya que por ejemplo existirán 295 direcciones por cada ser humano existente en el planeta.

La transición de IPv4 a IPv6 Para la mayoría de los usuarios su necesidad de conectividad radica en el acceso a la web, correo electrónico y a su conjunto de aplicaciones particulares (Facebook, twitter,etc.) y esta necesidad debe estar cubierta los 7 días de la semana las 24hs del día. Por lo tanto para realizar un buen despliegue de IPv6 en cualquier infraestructura de red, nos enfrentamos al reto de lograr la misma de manera que transparente para el usuario (o sea que no experimente ninguna suspensión en la conectividad a la red) y no como una

experiencia traumática. Es por ello que toda transición o migración de tecnología debe estar planificada sobre los siguientes tres pilares fundamentales:

1. Planificar el despliegue de manera precisa por fases.
2. Realizar pruebas sobre el diseño de la nueva infraestructura para evaluar el funcionamiento de éste y detectar cualquier problema antes de desplegar la nueva red.
3. Desplegar la red en el entorno de producción.

Una vez alcanzada la finalización del proceso de migración de la red a IPv6, seguramente estaremos en presencia de un escenario heterogéneo donde las redes con las cuales establecia conexión la red migrada siguen funcionando con la versión anterior. Es por ello que existen varias técnicas que posibilitan la coexistencia entre redes con distintas versiones de IP y asi lograr el aseguramiento del servicio a los usuarios.

La transición tecnológica como ciclo contrastante de la realidad En todo proceso de migración o transición de tecnología no se puede asegurar a priori que las fases ideadas no se vean a afectadas por fallos inesperados, es por ello que este tipo de proceso se realiza paulatinamente en entornos controlados y en ciclos de puesta en producción, testeo, mantenimiento y vuelta a producción. Estos ciclos contrastantes van generando conocimiento a posteriori de cada iteración como es propuesto por el empirísmo.

3.3 Aplicaciones Inspiradas en la Biología

Muchas de las líneas de investigación en la informática se han inspirado en la biología, en esta sección presentaremos las dos más relevantes y como estas se relacionan con el empírismo.

Algoritmos Genéticos *Un algoritmo genético (AG) es una técnica de programación que imita a la evolución biologíca como estrategia para resolver problemas. Dado un problema específico a resolver, la entrada del AG es un conjunto de soluciones potenciales a ese problema, codificadas de alguna manera, y una métrica llamada función de aptitud que permite evaluar cuantitativamente a cada candidata. Estas candidatas pueden ser soluciones que ya se sabe que funcionan, con el objetivo de que el AG las mejore, pero se suelen generar aleatoriamente. Luego el AG evalúa cada candidata de acuerdo con la función de aptitud. En un acervo de candidatas generadas aleatoriamente, por supuesto, la mayoría no funcionarán en absoluto, y serán eliminadas. Sin embargo, por puro azar, unas pocas pueden ser prometedoras -pueden mostrar actividad, aunque sólo sea actividad débil e imperfecta, hacia la solución del problema. Estas candidatas prometedoras se conservan y se les permite reproducirse. Se realizan múltiples copias de ellas, pero las copias no son perfectas; se introducen cambios aleatorios durante*

el proceso de copia. Luego, esta descendencia digital prosigue con la siguiente generación, formando un nuevo acervo de soluciones candidatas, y son sometidas a una ronda de evaluación de aptitud. Las candidatas que han empeorado o no han mejorado con los cambios en su código son eliminadas de nuevo; pero, de nuevo, por puro azar, las variaciones aleatorias introducidas en la población pueden haber mejorado a algunos individuos, convirtiéndolos en mejores soluciones del problema, más completas o más eficientes. De nuevo, se selecionan y copian estos individuos vencedores hacia la siguiente generación con cambios aleatorios, y el proceso se repite. Las expectativas son que la aptitud media de la población se incrementará en cada ronda y, por tanto, repitiendo este proceso cientos o miles de veces, pueden descubrirse soluciones muy buenas del problema.[9]

Redes Neuronales Una red neuronal es un método de resolución de problemas basado en un modelo informático de la manera en que están conectadas las neuronas del cerebro. Una red neuronal consiste en capas de unidades procesadoras, llamadas nodos, unidas por conexiones direccionales: una capa de entrada, una capa de salida y cero o más capas ocultas en medio.
Se le presenta un patrón inicial de entrada a la capa de entrada, y luego los nodos que se estimulan transmiten una señal a los nodos de la siguiente capa a la que están conectados. Si la suma de todas las entradas que entran en una de estas neuronas virtuales es mayor que cierto umbral de activación de la neurona, esa neurona se activa, y transmite su propia señal a las neuronas de la siguiente capa. El patrón de activación, por tanto, se propaga hacia delante hasta que alcanza a la capa de salida, donde es devuelto como solución a la entrada presentada. Al igual que en el sistema nervioso de los organismos biológicos, las redes neuronales aprenden y afinan su rendimiento a lo largo del tiempo, mediante la repetición de rondas en las que se ajustan sus umbrales, hasta que la salida real coincide con la salida deseada para cualquier entrada dada.

Experiencia como fuente del conocimiento Según el empirismo *"la experiencia es la única fuente del conocimiento"*.
Tanto los algoritmos genéticos como las redes neuronales basan su funcionamiento en la adquisición de experiencia. Los primeros a través de los ciclos evolutivos, mientras que los segundos por medio de la fase de entrenamiento. Por lo tanto las redes neuronales como los algoritmos genéticos se encuadran perfectamente como aproximaciones empiristas.

4 Conclusion

Se han examinado casos testigos de acuerdo a la conveniencia y cercanía con las corrientes racionalista y empirista, de manera que sea posible resaltar claramente las influencias en cada caso.

El racionalismo con un matiz netamente antropocéntrico, posiciona al hombre en un lugar central sin límites en el entendimiento del *todo*. El mecanicismo cartesiano reafirma esto y contribuye a la idea consecuente del hombre como creador (de mecanismos) o gran diseñador. Lejos de estar en el ocaso, esta idea ha estado subyacente desde la primera revolución industrial continuando en los refinamientos de los procesos industriales, en las ingenierías y en la joven vida de la profesión informática.

El método de desarrollo en cascada, tal como fue desarrollado en este documento, es una de las primeras formulaciones dentro de lo que en informática se llamó *ingeniería del software*. Este abordaje sufre de precariedad y rigidez, pero tiene asidero en que fue una de las primeras respuestas a lo que en los principios de la década del 70 se llamo *la crisis del software* [11], intentando mejorar los tiempos de creación del software, minimizar los costos, mejorar la verificabilidad y flexibilidad del software desarrollada. Esta primera versión de desarrollo en cascada, adoleció de una gran ingenuidad por parte de los especialistas que generaron el método. Cayendo eventualmente en la misma problemática que se intentaba superar. El método invierte una gran cantidad de energía en definir, pensar y diseñar el andamiaje de lo que después sería el sistema, postergando lo más posible las etapas de constrastación con la realidad. Una vez que el sistema funciona, se pasa a la etapa de mantenimiento en la cual se realiza la adecuación de lo ideado a la realidad.

Cabe destacar que el desarrollo en cascada es usado actualmente pero no en la forma básica y pura desarrollado en este documento. Nuevas versiones, modificaciones y adecuaciones al método de desarrollo en cascada existen. Minimizando así la precariedad y rigidez del primero, aunque no por ello deja de conservar un origen netamente racionalista.

El desarrollo empirista del siglo XVII continua el matiz antropocéntrico del racionalismo, sin embargo asume al hombre como alguien acotado, posicionándolo asi en el lugar de *aprendiz* de la realidad. La realidad es abordable hasta cierto punto y el hombre necesita censar/percibir de alguna manera los fenómenos del mundo. Esto es necesario para empezar la construcción de las implicancias de las actividades sucedidas en realidad, y así, poder formular modelos, ideas y nociones generales de la complejidad del mundo.

Tanto los algoritmos genéticos como las redes neuronales y las aplicaciones de simulación basan su funcionamiento en la adquisición de experiencia. Los primeros a través de los ciclos evolutivos, mientras que los segundos por medio de la fase de entrenamiento. En este tipo de escenarios el desarrollador queda relegado al lugar de observador/supervisor de un proceso autónomo, que se auto regula mediante el censado del contexto. En contraposición los abordajes imperativos quedan excluidos en estos escenarios. No sería factible definir a priori la actividad que *debiera* suceder dentro de este tipo de aplicaciones, ya que el objetivo de éstas es detectar un emergente mediante un gran número de iteraciones y la experiencia obtenida a lo largo del proceso.

En escenarios no-lineales como la implantación de IPv6 los especialistas asumen una óptica más intervencionista, dado que no es del todo claro, ni determinista,

el comportamiento del sistema. En los estadios tempranos de cada intervención es necesario censar la actividad y someter a un análisis exhaustivo los resultados. Posteriormente la intervención en sí, es llevada a cabo de acuerdo a las reformulaciones originadas en el análisis de la actividad percibida. No es difícil observar aquí las influencias netamente empiristas, ni las palabras de John Locke cuando señala que el conocimiento posible solo puede suceder a *posteriori*. Por último es importante destacar hasta qué punto estas corrientes de pensamiento están implicadas en la actividad informática. Aunque en la misma no se mantienen enfoques puristas como los presentados en este documento, el conjunto de actividades informáticas son entonces una yuxtaposición de estos abordajes. El surgimiento de nuevas técnicas para la resolución de problemas dentro de la disciplina, actualmente es una composición de ambas tesituras fijando etapas en las cuales toma un rol más preponderante una para luego cederselo a la otra en una etapa posterior.

References

1. RACIONALISMO Y EMPIRISMO EN LA LINGSTICA DEL SIGLO XVII: JOHN WILKINS Y PORT-ROYAL. Xavier Laborda Gil
2. El discurso del Metodo Rene Descartes. http://www.librosmaravillosos.com/metodo/index.html
3. Analisis de Sistemas- Juan Bravo Carrasco- Cap 4.
4. Complexity and Postmodernism Paul Cilliers Chapter 1
5. Systems simulation: the art and science - Shannon, Robert; Johannes, James D. (1976). IEEE Transactions on Systems, Man and Cybernetics 6(10). pp. 723-724.
6. Connectivity Probability of Wireless Ad Hoc Networks: Definition,Evaluation, Comparison - Tatiana K. Madsen, Frank H. P. Fitzek and Ramjee Prasad
7. A SURVEY OF DATA REPRESENTATION STANDARDS http://tools.ietf.org/html/rfc971
8. The Recommendation for the IP Next Generation Protocol http://tools.ietf.org/html/rfc1752
9. Algoritmos genéticos y computación evolutiva: Adam Marczyk (2004) http://thegeek.org/docs/algen/
10. Red neuronal artificial https://es.wikipedia.org/wiki/Red$_{n}euronal_{a}rtificial$
11. The humble programmer by Edsger W.Dijkstra http://www.cs.utexas.edu/ EWD/ewd03xx/EWD340.PDF