



## **VI Congreso nacional de la Asociación Mexicana de Estudios del Trabajo**

<http://www.uaq.mx/amet/home.html>

### **Formulación de nuevos paradigmas en los estudios del trabajo**

**Eje temático:**

**Competencias, sistemas educativos y de aprendizaje**

**Mesa de Trabajo:**

**El trabajo simbólico más allá de la sociedad del conocimiento**

**Ponencia:**

**Trabajadores cognitivos en la Industria del Software  
Hacia un concepto ampliado de trabajo simbólico**

**J. Guadalupe Rodríguez Gutiérrez\***

**Universidad de Sonora-Nogales  
Universidad Autónoma Metropolitana-Iztapalapa**

**Querétaro, México,  
a 23 de Mayo de 2008**

#### **Resumen**

La producción simbólica es una de las características del modo de producción capitalista de principios de siglo y su característica principal es la circulación y consumo de signos y símbolos en el proceso de trabajo. Coexisten diversos tipos de producción simbólica como es la: cognitiva, de símbolos, emotiva, etc. La producción simbólica rompe con el pasado fordista-taylorista al participar un tercer jugador en el proceso de producción: el consumidor final y/o cliente.

**Palabras claves:** Subjetividad, producción simbólica, Software, Lenguajes de programación.

## 1.- Agotamiento del concepto tradicional de trabajo en la producción simbólica

Para los posmodernos la característica principal del capitalismo moderno es la fluidez de la información y el conocimiento en el contexto general de la economía del conocimiento<sup>1</sup>; la particularidad de ésta economía –según los posmodernos- es que no posee más la centralidad del trabajo como generadora de valor (Castells e Himanen:2002)<sup>2</sup> y que el nuevo componente clave es inmaterial: información y conocimiento como dispositivo central en la generación de valor (Hardt y Negri:2000)<sup>3</sup>. Otros enfoques incluyen la interacción entre agentes del tipo red, donde el proceso innovador está dado por las relaciones en redes que establecen los agentes sociales y locales a través de los encadenamientos y efectos de “derrama” que se generan a partir de sus prácticas productivas (Yoguel: 2003)<sup>4</sup>. Otra particularidad es que se registran nuevas formas de organización del trabajo tendientes a reducir la división que separa el trabajo intelectual (concepción) del manual (ejecución) (Luna:2003)<sup>5</sup>; brecha que, supuestamente es menor debido a la mayor presencia de un nuevo tipo de alineación del trabajador inmaterial, que se configura como carburante de las nuevas formas de trabajo del capitalismo moderno (Lazzarato y Negri: 2001)<sup>6</sup>. De tal forma que según Hardt y Negri:2000<sup>7</sup> y otros, se está hoy día frente a un *nuevo tipo de alineación* del trabajador, circunstancia que es intrínseca al trabajo inmaterial. Alineación que estaría dada por la actividad productiva inmaterial, donde es imposible distinguir trabajo y satisfacción subjetiva del gusto por el trabajo; además que en la actividad productiva actual la importancia recae en el saber social general (intelecto general) sea éste producto del desarrollo científico o de la interacción social. Este saber social general ha ido modificando el ciclo de producción, redefiniendo la relación producción/consumo que deja de ser fabricación de grandes cantidades de objetos estandarizados en serie (crisis del modo de producción del Fordismo/Taylorismo) para convertirse en un proceso que, desde el diseño del producto

---

<sup>1</sup> Para una lista extensa de la retórica de la nueva economía basada en el conocimiento véase Business Week “21st century capitalism” 12 de Septiembre de 1994; Castells, Manuel y Pekka Himanen (2002) El estado del bienestar y la sociedad de la información, Alianza, Editorial, Madrid, 215 pp.; Organización Internacional del Trabajo (2002) Aprender y formarse para trabajar en la sociedad del conocimiento, Ginebra, Informe IV. 136 pp.; Banco Mundial (2003) Aprendizaje permanente en la economía global del conocimiento. Desafío para los países en desarrollo; Editorial Alfaomega, pp. 152.

<sup>2</sup> Castells, M. y P. Himanen, El estado del bienestar y la sociedad de la información, Alianza, Madrid, 2002

<sup>3</sup> Hardt M. y A. Negri, Imperio, Ciruela, Argentina, 2004

<sup>4</sup> Yoguel, G. “Innovación y aprendizaje: las redes y sistemas locales”, en Yoguel, G. y Boscherini, F., (2001), “El desarrollo de capacidades innovativas de las firmas y el rol del sistema territorial.”, Desarrollo Económico, 2003;

<sup>5</sup> Luna, M., (Coord.) Itinerarios del conocimiento: formas dinámicas y contenido. Un enfoque de redes, Ed. España (Anthropos) México (Universidad Autónoma Metropolitana Izt.) pp. 19-50. 2003

<sup>6</sup> Lazzarato M. y A. Negri, Trabajo inmaterial. Formas de vida y producción de subjetividad, DP&A Editora, Río de Janeiro, 2001 Disponible en: <http://www.rebellion.org/libros/TrabajoInmateria011202.pdf>. Acceso en: 05/09/07

<sup>7</sup> Hardt M. y A. Negri, op.cit.

considera las prerrogativas del consumidor a través de un proceso fluido de información entre producción y consumo. La producción inmaterial no simplemente acerca a consumidores y productores, sino que establece nuevas relaciones sociales de producción, donde la materia prima -desde una perspectiva más amplia- son las condiciones subjetivas de los consumidores. La subjetividad se produce ya no como instrumento de control social, sino como condición de acumulación del nuevo ciclo del capital productivo.

Este discurso, que De la Garza:2008<sup>8</sup> denomina para-posmoderno es construido a partir de una serie de confusiones. La primera hace referencia a que se da por hecho que en la economía del conocimiento existe un acceso y cobertura global tecnoproductiva de las tecnologías de la información y la comunicación que a su vez se explica por un contexto amplio de la revolución de la inteligencia<sup>9</sup>; donde la materia prima que estimula la generación de conocimiento es la alineación de la subjetividad del obrero, limitando así lo inmaterial a la subjetividad de los individuos en un marco racionalista, como si la subjetividad no estuviese compuesta por variables aleatorias, fortuitas, moldeables, contingentes, manipulables. Una segunda confusión es el concepto de producción inmaterial que lo hacen sinónimo de *producción industrial de servicios* (Zarifian:1999)<sup>10</sup> es decir, en el intento de explicar esta equivalencia, incurren en una tercera confusión, dar cuenta de una progresiva convergencia entre esfera de producción y consumo, a través de argumentos estructurales, racionalistas; señalando la creciente necesidad de suministrar a los clientes de servicios a la “medida”. Consideramos que estas confusiones –entre otras- obscurecen y ocultan una de las particularidades del capitalismo moderno: la cada vez mayor presencia de productos simbólicos y proceso de trabajo de creación de símbolos (para una mayor ampliación de estas confusiones léase De la Garza:2008 op.cit.) sin que ello implique necesariamente una nueva economía, o un nuevo estadio de la producción capitalista, quizá el mejor ejemplo es la persistencia de la brecha digital, antes denominada brecha tecnológica, que hoy sólo cambio de apellido, o bien la crisis alimentaría de esta segunda década del siglo XXI.

---

<sup>8</sup> De la Garza, Enrique, et.al. 2008 Crítica de la razón para-posmoderna (Sennet, Bauman, Beck). Revista Latinoamericana de Estudios del Trabajo, Argentina. Segunda época, año 13, número 19, 2008. pp. 6-37.

<sup>9</sup> "Revolución de la Inteligencia". En la actualidad, 85% de todos los científicos que han vivido a lo largo de toda la historia están vivos y cuentan con herramientas más avanzadas y mayor creatividad. Ello ha conducido a que la tasa de cambio científico y tecnológico sea más rápida que en el pasado. Actualmente el conocimiento científico se duplica aproximadamente cada 5 años. Las áreas donde están surgiendo más innovaciones tecnológicas son energía nuclear, informática, robótica, biotecnología, telecomunicaciones y ciencias del espacio. <http://es.wikipedia.org>

<sup>10</sup> Zarifian, Philippe. El modelo de competencia y los sistemas productivos. Montevideo: Cinterfor, 1999. Papeles de la oficina técnica, OIT.

Hacia fines del siglo XX cobran relevancia cierto tipo de industrias que producen bienes simbólicos que pueden objetivarse o permanecer subjetivos, como por ejemplo el consumo del espectáculo y el entretenimiento, la producción de una obra de teatro o consumo de un film en el cine, el consumidor final sólo percibe subjetividades de satisfacción, descontento o placer. Aquí, como en otros servicios, por ejemplo el asistir con el médico, en el restaurante etc. el proceso de producción no se concreta a una separación material de la producción por un lado y el consumo por el otro, no se delimitan las fronteras entre proceso y consumo, entre demanda del producto y oferta del mismo; sino que es en el acto mismo del consumo donde se potencia y precisa el servicio mismo. De la Garza, señala a este tipo de producción simbólica definida como aquel servicio-producto que “no existe separado de la propia actividad de producir y que de manera ideal comprime las fases económicas tradicionales de producción, circulación y consumo en un solo acto” la objetivación no se separa del consumo y el producto es exclusivamente simbólico, de tal forma que su componente material resulta secundario. De esta manera “se complejizan, así, las relaciones sociales de producción al hacer intervenir a un tercer sujeto de manera inmediata en el proceso de producción junto al trabajador y el patrón, cuando es trabajo asalariado” (De la Garza mimeo, 2007:7). También puede haber una producción simbólica objetivada que puede almacenarse y revenderse posterior a la etapa de producción, como es el desarrollo de software, el diseño de páginas web, entre otros. Este tipo de producción simbólica que se objetiva fuera del espacio de producción y consumo, rompe con varias rigideces del ciclo productivo de la economía industrial clásica:

- Producción eminentemente de símbolos, que pueden ser del tipo cognitivo, morales, estéticos, emotivos, etc.; que consiguen objetivarse o no. En esta producción se sucede una yuxtaposición del ciclo *Producción – Circulación – Consumo* comprimiéndose en un solo acto.
- Las relaciones sociales pueden complejizarse en el piso de producción, ya que las tradicionales relaciones trabajador-patrón, pueden verse alteradas (como en la producción de software a la medida) por la presencia inmediata del cliente al principio del proceso de trabajo, como hacia el final del mismo (consumidor final).
- En el proceso de trabajo simbólico, es confuso separar las fronteras entre dimensiones objetiva y subjetiva y el acto mismo de la creación del objeto. De la Garza, propone que “la objetivación se da de manera automática en otro sujeto, el cliente o usuario y no en un objeto separado de los dos.” (De la Garza mimeo, 2007:4). Pero también, puede haber

una producción puramente de símbolos y éstos estar objetivados o no. Por ejemplo los programas informáticos como el software, las paginas web, diseño de juegos electrónicos, especulación financiera, películas totalmente animadas por computadora, entre otros.

- Mientras que en la manufactura tradicional el producto es un objeto externo al trabajador y el patrón lo ofrece al mercado, depende del juego de oferta y demanda (teoría neoclásica); en la producción de bienes simbólicos, esto no ocurre, el consumo es casi inmediato a la producción y éste consumo depende de una serie de subjetividades de quien consume el producto. Aunque en la producción de símbolos objetivados (como el software y las paginas web, por citar algunos) las fases de producción y consumo vuelven a traslaparse.
- Si bien es cierto la actividad laboral sigue siendo de interacción a veces entre sujetos cara a cara y en otras ocasiones no es así, lo cual no necesariamente significa que es una condición en el proceso de trabajo inmaterial, en especial en el desarrollo de software que puede ser pantalla-pantalla, en tiempo virtual o fuera de la jornada de trabajo (yuxtaposición de los espacios laborales y espacios de vida del trabajador).

En este punto podemos presuponer que la actual fase de producción capitalista la producción simbólica resulta relevante sea porque lo que se produce son solo símbolos (por ejemplo los programas informáticos como el software, los juegos electrónicos, etc.) o por la importancia en la producción material de la subjetividad del cliente. Intrínsecamente al concepto de producción simbólica coexisten diferentes tipos de trabajos simbólicos:

- *Trabajo cognitivo*, se entendería por aquellas actividades en las cuales predomina el aspecto intelectual, reflexivo. Por ejemplo, la actividad académica, el periodismo, la abogacía, el diseñador gráfico, el diseñador virtual, programas de software, etc.
- *Trabajo emotivo*, estaría compuesta por diversas actividades en las cuales los sentimientos, las emociones, las pasiones, etc. están presentes en mayor medida, (Bulton:2006, Vigotsky;2004 [1934], Girard:1975), por ejemplo las empleadas de mostrador, los sobrecargos en aviación; empleados bancarios, profesionistas de radio, prensa y televisión, empleados de call center, etc.
- *Trabajo estético*, formarían parte aquel conjunto de actividades que se transmiten a través de signos abstractos como el sentimiento, las formas, expresiones o composiciones de notas musicales, libretos de film, literatura, etc. por ejemplo, los

compositores e interpretes, los dramaturgos, escritores y literatos, arte fotográfico, pintura, escultura, danza, etc.

- Entre otro tipo de trabajos simbólicos.

Como ya explicamos no toda producción simbólica es inmaterial. Es importante distinguir dos espacios muy generales: Trabajo simbólico subjetivado y trabajo simbólico objetivado:

- *Trabajo simbólico subjetivado*: Se comprende por un tipo de trabajo que se subjetiva en el consumidor, donde se yuxtapone y comprime la producción-circulación y consumo en un solo acto. El consumidor es el tercer actor que participa –en menor medida- en la concreción del producto y éste se consume en el acto mismo que se produce, se concreta en la subjetividad de quien le consume. Por ejemplo, los restaurantes con venta a la carta, el consumo de un concierto musical en vivo, el médico, el académico, etc. Donde el producto se concreta a través de las representaciones estéticas, sentidos del sabor, del gusto, olor, color etc.
- *Trabajo simbólico objetivado*: Se hace referencia a un tipo de trabajo simbólico que se objetiva fuera del consumidor, de tal forma que la esfera de producción-circulación-consumo no es suficiente para explicar la presencia de un tercer actor (el consumidor) al interior del proceso de trabajo. Por ejemplo el diseñador de páginas web, el desarrollo de software, entre otros, donde si bien es cierto el cliente participa como productor e incluso como diseñador del o los requisitos/características del producto, éstos se objetivan fuera del cliente. Por ejemplo, el diseño de una pagina web y la publicidad virtual (formas, esquemas, colores, etc.) ó los programas de software a la medida del cliente, si bien se entregan al cliente en un CD o USB o en cualquier otro medio electrónico o virtual, éstos programas se “virtualizan” en otros espacios que no son del cliente, por ejemplo la industria de la piratería.

El esquema positivista estructural de dividir lo cognitivo y la materia, es criticado por Godolier (1987) para quien la separación mente-materia no existe, por el contrario señala que la cultura del trabajo es el hilo de intersección entre símbolo y materia; Para Burawoy (1989) la cultura del trabajo está compuesta por el conjunto de conductas, procedimientos formales e informales, significados y hábitos de organización en las que están inmersos las relaciones laborales; Gintis (1983) señala que la cultura laboral como la cultura empresarial son producto de la interacción entre sujetos inmersos en las relaciones laborales, y, éstas

están constituidas tanto por elementos cognitivos (caracterizaciones de las ocupaciones y saberes laborales) como por elementos no cognitivos (grupos de referencia normativos)<sup>11</sup>. Para Reygadas (2000) la dimensión simbólica y la dimensión productiva se intersecan en la cultura laboral. La separación entre concepción y ejecución de Taylor y Ford, encuentra sus límites en estos contextos, donde se debe analizar la influencia de lo simbólico en lo material y viceversa, no están dissociadas, por lo contrario se vinculan, se entrelazan y se reconfiguran mutuamente, ya que cultura incluye signos, símbolos que se refieren a conocimientos, informaciones, valoraciones, emociones, sentimientos, ilusiones, utopías, etc., y éstas no existen externas al individuo, por el contrario son expresión de una acumulación de normas sociales (Varela:1994 pp. 3-4, citado en Reygadas, 2000). Para el caso de los programadores de software hemos detectado operaciones subjetivas que configuran una serie de características normativas, acumuladas como códigos sociales, por ejemplo los valores, las representaciones, las ideas, las normas y hábitos contextuales que estructuran la acción del programador y es constreñida esta acción por las configuraciones subjetivas del programador frente al trabajo. La simbiosis entre lo físico y lo simbólico, lo material e inmaterial están subjetivadas entre las interacciones individuales y colectivas de los agentes, instituciones y actores que intervienen en el proceso de trabajo del software (Cliente, diseñador, programador senior, programador junior, tester, documentador, etc.)

Para el caso de los trabajadores cognitivos del software, se trata de una producción simbólica objetivada, donde la objetivación posee varias fases o etapas, sin que ello represente una división del trabajo al estilo fordista-taylorista. Podemos, sintetizar las fases y etapas del proceso de trabajo en el desarrollo del Software en las siguientes Etapa 1: *El proceso de desarrollo*: a) El cliente contacta a la empresa Desarrolladora. Se llevan a cabo una serie de entrevistas para determinar la lista de requisitos y estipular las fechas de entrega de los módulos que integran el Sistema; b) Una vez convenida la licitación de Requisitos del Sistema, se pasa al Diseño, configuración y elección del *ciclo de vida* del desarrollo del Software. Esta información se entrega al Líder de proyecto o programador líder; c) Proceso del Trabajo. El Líder de proyecto, es el que asigna las tareas a los programadores (Junior y Senior). Una vez que el programador concluye las rutinas, son

---

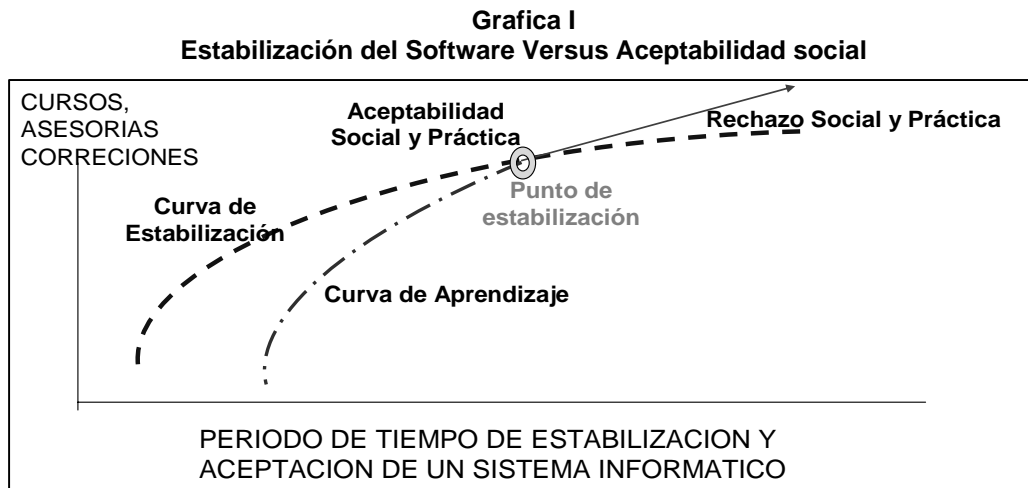
<sup>11</sup> Godolier Maurice (1989), *Lo ideal y lo material*, Madrid, Taurus Humanidades. Gintis. Herbert (1983) "La naturaleza del intercambio laboral y la teoría de la producción capitalista"; en Tohara, L. (Comp.) *El mercado de trabajo: teorías y aplicaciones*, Madrid, Alianza Universidad, 1986, p. 176; Además léase Krotz, Esteban (1993), *la cultura Objetivada*, México, Universidad Autónoma Metropolitana – Iztapalapa;

revisadas por el Tester, quien hace pruebas de calidad, “estresa” al sistema con la finalidad de que se exhiban problemas internos de configuración. *Etapa 2: El proceso de implementación.* Se inicia una vez que es aprobado el software que se entrega al cliente; d) Implementación en campo y prueba con usuarios Finales. Es el periodo de tiempo en el cual se convergen dos momentos: **La curva de estabilización** del sistema informático y la **curva de aprendizaje** de los usuarios finales. En el cruce de ambas podemos decir, que se acepta o se rechaza el programa informático desarrollador (ver grafica I). En la curva de estabilización del Software, es el tiempo en el cual se llevan a cabo toda una serie de revisiones técnicas, desde aquellas que verifican que tan “amigable” es el interfaz grafico, pasando por el de operabilidad con el hardware existente e interoperabilidad con otros sistemas informáticos (en el caso de existir); por ejemplo si interopera con una red externa, Internet, Intranet, si existe comercio electrónico, etc. Simultáneamente la curva de aprendizaje del Software, se transcurre cuando el usuario final consume, ejecuta el software en su ámbito de trabajo, donde puede haber problemas de operabilidad con el hardware, errores de configuración del software al momento de instalarlo en las computadoras del cliente. Sin embargo, intrínsecamente existe una resistencia por parte del usuario final, resistencia que se puede traducir en boicot al uso del software nuevo, en negación implícita o rechazo social del Software que se implementa. Sin embargo, también coexiste un rechazo en el uso del software desarrollador. La ingeniería de la usabilidad señala que 30% de los errores en los sistemas informáticos no radica en los algoritmos desarrollados, sino en el conjunto de iconos contextuales, formatos, colores y exposición simbólicas no son claros, comprensibles para los usuarios finales. Ambas curvas, tanto la de Aprendizaje como la de estabilización pueden converger antes en la empresa, sin embargo, dependerá del grado de aceptabilidad social o rechazo en la practica del Software al interior de la empresa, que prolongara el punto de estabilización del Software, o en su defecto ello se puede traducir en un rechazo al programa del software por parte del usuario final ( ver grafica I).

Esta contingencia típica en la industria, es con el fin de señalar el grado de intervención y dependencia que el proceso de trabajo del Software se supedita al uso eficiente por el lado del cliente y usuarios finales. Es decir, si bien es cierto en el proceso de trabajo del Software se culmina el programa mismo, este solo co-objetiva su potencialidad cuando esta operando en las instalaciones del cliente, cuando es aceptado técnica y socialmente el Software. Por tanto tenemos que el desarrollo de un sistema informático, no se circunscribe sólo al periodo



que comprende al proceso de generación del conjunto de símbolos que dan origen al Software al interior del proceso de trabajo, sino que tiene que ver con dos procesos simultáneos, sincrónicos posteriores al proceso de trabajo.



FUENTE: Elaboración propia en base a entrevistas.

Podemos establecer algunas características del trabajo cognitivo en la producción de software a la medida: En el desarrollo existe una relación triádica clientes, productores y trabajadores; El producto cognitivo puede o no objetivarse fuera del productor y el cliente; Está compuesto por signos y símbolos; Existe una presión cognitiva en todo el proceso; El cliente señala una lista de requisitos –a veces directamente, en otras indirectamente- que debe cumplir el producto solicitado; El productor presiona cognitivamente a los trabajadores, que sean creativos y resuelvan de la mejor manera los requisitos que solicita el cliente; Las habilidades y destrezas, la colaboración y el juego, son mediados por conflictos y resistencias por el poder de decidir cual es el “mejor camino” en el desarrollo de los símbolos; La incertidumbre de cumplir o no con los requisitos está presente; El cliente es un sujeto activo en el diseño y requisitos del producto; El producto objetivado representa una porción significativa de conocimiento, el cual se enfrenta a dos dimensiones: convertirse en conocimiento codificado (manuales de procedimiento) o bien disolverse, fragmentarse, acumularse entre los individuos como conocimiento tácito; El producto se puede almacenar virtualmente, copiarse, optimizarse el uso, etc. El costo de reproducción tiende a cero. Es mínimo comparado con el primer producto desarrollado.

## **Un Trabajo centrado en el conocimiento: ruptura taylorista**

el proceso de trabajo en el desarrollo de un programa informático podríamos definirlo como *“un proceso de trabajo en el cual la materia prima son diferentes tipos de símbolos y el resultado es un conjunto de signos que se crearon a partir de descripciones, representaciones y significados”*, proceso que estaría conformado por cuatro grandes espacios no rígidos, sino flexibles y traslapados, con ambigüedad y elasticidades: a) La conceptualización de los requerimientos o necesidades del cliente, donde éstas conceptografías hacen referencia al conjunto de instrucciones que debe contener el proyecto informático a desarrollar; b) La formalización de los requerimientos en el diseño del proyecto son aprehendidas y modificadas por el analista, arquitecto o programador en un conjunto de instrucciones; c) El procesamiento de datos, es la transformación de las instrucciones del diseño en una serie de consideraciones lógicas (algoritmos) que resuelvan los requerimientos o necesidades del sistema; d) Implementación, es la puesta en marcha del programa desarrollado en el ambiente informático del cliente. La aceptación social o técnica del programa por parte del usuario final la abordaremos mas adelante.

Las primeras dos fases, por lo regular se consideran una sola, dependiendo del tipo de empresa y una vez que se convinieron los requisitos entre cliente y proveedor, sobreviene la etapa del diseño es presentado por el Arquitecto de Software, Ingeniero de Software, o bien por un programador que este al frente del área de diseño, es cuando podemos decir que se inicia el proceso de trabajo del programador de software. Las dos siguientes fases es el punto en el cual, la lista de requerimientos se transforman en tareas específicas que los programadores transforman en algoritmos que conforman el módulo, dando por resultado una tarea o comando que prescribía un módulo. Proceso que no es espontáneo, por el contrario, esta inmerso en una serie de acciones colectivas, individuales, virtuales, emocionales, de conocimiento formal e informal, experticia, habilidades cognitivas, destrezas o “timing cognitivo”,<sup>12</sup> etc. Independientemente del método que se utilice en el desarrollo y presentación del diseño el desarrollo de un sistema exige que, quienes participan se comprometan con los objetivos del proyecto. En este proceso de identificación para con los tiempos y tareas del proyecto, no solo deben tomarse en cuenta las habilidades

---

<sup>12</sup> Entenderemos por esos momentos de “timing cognitivo” de los programadores para resolver un problema. Lo contrario sería lo que en el argot de programación se conoce como “me cicle” que hace referencia a que el programador NO resuelve un problema.

profesionales y la fiabilidad cognitiva del programador, también debe poseer destrezas personales para comunicarse, tener un carácter disciplinado, y atributos importantes para el intercambio de ideas, ya que un proyecto de software requiere de un intenso intercambio de información, que en muchas ocasiones se desarrolla bajo condiciones de incertidumbre.

Ahora bien, las fases aquí señaladas, son construidas con fines de explicación, porque en el proceso de trabajo, las fronteras entre una y otra se desdibujan, interactúan entre una y otra. El cliente puede intervenir o ser llamado en algunas coyunturas de la formalización o procesamiento de datos, o bien el programador negocia las instrucciones del proyecto con el diseñador o viceversa. La toma de decisiones en la procesamiento de datos si bien es cierto es individual, no está exento de investigar fuera del proceso de trabajo, de interactuar con otros programadores frente a frente o pantalla a pantalla, en tiempo real o virtual. Sin embargo, persiste la toma de decisión individual en la estructuración o coherencia final del código. Así como la decisión individual de la calidad en la explicación del procedimiento reflexivo del código (comentarios al margen del código complejo, que no sea muy claro), así como de la documentación del procedimiento del módulo (acotaciones del tipo administrativo para darle mantenimiento posterior al módulo). Sin embargo es importante acentuar que las fases conceptualización/formalización y por otro lado procesamiento de datos/Implementación no es rígida dicha separación, las fronteras no están marcadas o estructuradas, por el contrario el tránsito del programador, el gerente, analista y desarrollador están invisibilizadas y están embebidas en una *constelación de relaciones subjetivas* están representadas por un conjunto de arreglos, consensos; conflictos, resistencias, relaciones de poder, ya sean individuales o colectivos que se inscriben, producen y reproducen formal e informalmente en las interacciones cotidianas al interior como al exterior del proceso de trabajo. Éstas representan la oportunidad de concebir, ejecutar y generar relaciones de poder y paralelamente se crean las tareas requeridas.

Este conjunto de *constelaciones de interacciones subjetivas* se suceden en un conjunto de dimensiones subjetivas, embebidas en espacios claroscuras, opacos, donde podemos distinguir cuando menos tres tipos de subjetividad: La primera que denominamos *subjetividad creativa*, tiene que ver con la toma de decisión individual y en otros casos colectiva, con la intencionalidad de indagar innovadores procesos al problema planteado o simplemente seguir con “lo que me ha dado éxito”; con los sentidos estéticos o reflexivos del

código desarrollado (simple o complejo), con el sistema de representaciones y cumplimientos de arreglos formales e informales para cumplimentar con los requerimientos solicitado en el tiempos y complejidad acordada, con la búsqueda de nuevos métodos o repetir procesos, etc. La *subjetividad signica*<sup>13</sup> esta representada por el esfuerzo reflexivo, por la contingencia cognitiva que significa procesar un algoritmo. Este procedimiento cognitivo comprende una serie de *interacción de pensamientos, movimientos reflexivos* que pueden ser producto de relaciones sociales o individuales, de conversaciones con los otros en tiempo real o virtual, frente a frente o pantalla a pantalla; con un experto o con libros, en tiempo de trabajo o en el espacio de vida extra-laboral. La tercera, es subjetividad que se objetiva. Nos referimos a la influencia de un programa que se objetiva fuera del espacio del proceso de trabajo y que transforma el área donde se implementa.

Es importante considerar que estas subjetividades están presentes en las distintas fases que señalamos del proceso de trabajo. La naturaleza de la *subjetividad creativa*, se recrea y establece a partir del primer contacto (entrevista cliente-analista o programador) donde el cliente solicita el desarrollo de un software a la medida, esta relación esta inmersa en un contexto subjetivo que puede derivar en incertidumbre, porque las interpretaciones de las necesidades del cliente pueden ser próximas o asertivas al problema, pero también pueden estar alejadas de las necesidades reales del problema. La naturaleza de la *subjetividad signica*, reside en la configuración reflexiva del conjunto de signos que componen los lenguajes de programación (existen en la actualidad mas de dos mil lenguajes de programación) en el que se decide desarrollar el programa informático sea o no un lenguaje de programación optimo. De ambas subjetividades, podemos señalar que el proceso de trabajo es un conjunto subjetivo de crear “símbolos a través de signos”, proceso que está embebido de subjetividades que debemos definir para así abordar las particularidades que acontecen en la comunidad cognitiva de los trabajadores de software.

El resultado es un conjunto complejo de “símbolos creados a partir de signos” que nos son otra cosa que un conjunto de textos sígnicos (algoritmos) que solucionan e innovan un proceso solicitado por el cliente o usuario final. La innovación que el cliente posee en sus manos, -por llamarlo de alguna manera- representa un conjunto de iconos que se esparcen

---

<sup>13</sup> El algoritmo como texto signico. Bajtín, M, (2005, 1982) Op.cit.

en la pantalla del usuario final o cliente una vez puesto en operación, es hasta este momento en el que abordamos la tercera dimensión como un *proceso subjetivo que se objetiva* independiente al proceso de trabajo. Se caracteriza por objetivarse en la pantalla, ya sea en la del cliente/usuario final o en cualquier otra pantalla independiente del cliente; el programa de software a la media posee la característica de objetivarse n número de veces, después de haber salido del proceso de trabajo; el costo del segundo programa objetivado tiende a cero (el costo de copiarlo en un cd ó enviarlo por correo electrónico a las instalaciones del cliente que lo adquirió).

Debemos de estar pendientes, si en el sector servicios se implementan medidas de estandarización (Macdonalización, Walmartización, entre otras) y en que sentido, se aleja o acerca a los tradicionales procesos de una mayor intervención de la gerencia, ya sea a través de tecnología blanda (organización del trabajo en el piso de producción, que para el caso de los programadores de Software, sería una organización a nivel de pantalla de la computadora) o tecnología dura (Hardware) se esperaría mayor independencia entre las partes que integran el proceso de trabajo, una mayor separación de las características de la fuerza de trabajo individual, una mayor separación entre las fases que integran el proceso de producción, con el fin último -siguiendo a Braverman- que la descualificación se diese en la pantalla del programador de Software. Por ejemplo, para algunos la solución de la aflicción del software se resolverá en la medida que se implementen herramientas y métodos que resulten en una mayor estandarización y control en el proceso de producción del software, es decir que la descualificación en este trabajo, posiblemente este basada en una “administración científica de los tiempos y pensamientos” que pretende la Ingeniería del Software y las demás ingenierías que giran alrededor, como la Arquitectura del Software, Teoría Cliente-Ordenador, Teoría de la Usabilidad, Teoría de la Ergonomía del Software, etc.

Sin embargo, existe un conjunto de límites a las intenciones de estandarizar la producción del software:

- Participación proactiva del cliente en distintas fases del proceso de trabajo.
- Incertidumbre en el proceso.
- Ejecución de las LDC (Líneas de código) acorde a las Destrezas, habilidades y, un conjunto de reglas informales entre los integrantes de los equipos de trabajo.

- Tomas de decisión in situ, en torno a que método es el más adecuado en la ejecución de rutinas y subrutinas de las LDC que integran un módulo.
- La toma de decisiones in situ combina un conjunto de arreglos formales e informales. Por ejemplo, el nivel de experticia de quien propone una solución, las destrezas y habilidades cognitivas de quien propone, es decir, no necesariamente es experto, pero posee un talento en el discernimiento de la solución requerida.
- En el proceso de trabajo del software la comunicación del como y el porque se desarrollo determinadas rutinas de LDC del sistema, está basado también en una doble operación: Por un lado, el aspecto formal de la documentación del algoritmo desarrollado (conocimiento codificado); sin embargo, esta descripción carece de una serie de explicaciones detalladas, reflexivas. Así, coexiste una transmisión de información de contenidos lógico-deductivos, cognitivos, los cuales no se transmiten hacia el final de la elaboración de las LDC, como es el caso en la documentación del algoritmo, sino que se va trasmitiendo en el desarrollo mismo del código. Esto ocurre entre el Programador Junior y el programador Senior. Entre el programador responsable de un modulo y el líder o administrador del proyecto.
- En la producción de software sigue habiendo una “flexibilidad cognitiva” y la formación de “arreglos sociales de participación”

Podemos proponer que coexisten varias polémicas en el proceso de trabajo del software:

- Conformación de un conjunto de *constelación de relaciones subjetivas* que se construyen formal e informalmente; representan intencionalidades y conflictos que pueden ser individuales o colectivos. Estas son inherentes al proceso de trabajo: subjetividad creativa, subjetividad signica y subjetividad que se objetiva.
- Alto grado de integración del trabajo social en la pantalla del programador, que limita la racionalización de tareas al estilo taylorista. Aún con la fragmentación de tareas en el proceso de trabajo del software: Conceptualización (gerente-cliente); formalización (analista/arquitecto/desarrollador); Procesamiento de datos (Programadores, Tester, Documentador) e implementación (programadores, gerente, vendedor).
- *rigidez técnica* en la reorganización de los programas informáticos del software en ciclos cortos de cambio tecnológico;
- *inflexibilidad en la socialización del aprendizaje*, es decir presencia de un ciclo relativamente largo para familiarizarse en el uso y manejo del software. En otras

palabras, cuando se implementa un nuevo software, el usuario final se opone, se resiste en el uso y aprendizaje del nuevo software.

- *Flexibilidad cognitiva*: Entendiendo como aquellas acciones objetivas y subjetivas del tipo reflexivas, de rutina o cognitivas que llevan a cabo los agentes que tomas decisiones individuales o colectivas.
- *Arreglo social de la participación*: comprendemos aquellos intercambios sociales frente a frente, pantalla a pantalla en tiempo real o virtual, de experiencias, habilidades y destrezas que permiten al programador solucionar o imaginar una posible solución algorítmica.
- *Timmig Cognitivo*: Comprendemos una capacidad cognitiva que no esta en función de experiencias, estudios formales o intercambio de conocimiento. Simplemente la excepcionalidad en la solución de problemas (“no todos los programadores pensamos igual”)

Por ultimo señalamos que el proceso de trabajo del software está contextualizado por la incertidumbre, si bien es cierto el diseño del software se planifica, donde deben destacarse tiempos de finalización, fechas de implementación del programa (ejecutable para el usuario final), también debe agregarse una estimación de los costos de desarrollo, regularmente estimado en horas-hombre. Sin embargo, estos tiempos, fechas y costos de programación no son considerados eficientemente. La ingeniería del Software no ha logrado establecer metodologías o métricas de calidad que sean eficaces, que contemplen retrasos, errores en diseño, fallas del sistema -tanto en desarrollo como en implementación- (Pressman:2002). Otros investigadores, señalan que el conjunto de fallas y errores -considerado en la presente como “aflicción del software”- es un conjunto de incertidumbres características del riesgo en el desarrollo de un programa informático. Este riesgo forma parte del proceso de trabajo que varios métodos consideran. El riesgo forma parte del proyecto (Pressman:2002) traduciéndose en costos que deben ser estimados. Para Zahran (1998) el desarrollo de software debe comprenderse como una tarea continua de desafíos continuos. La respuesta a esta aflicción del software, por parte de académicos y empresarios para reducir al minimo el riesgo e incertidumbre que ello implica, se instituye la Ingeniería del Software -sobre todo la norteamericana- con un conjunto de metodologías para verificar procesos de calidad, gestión de proyectos, análisis formal de la arquitectura del software, conceptos y principios del diseño, configuración de software, métricas de prueba, etc., sin embargo, a más de dos

décadas de existencia de la Ingeniería del Software, el proceso de trabajo continúa embebido en riesgos de incumplimiento de los tiempos, fechas y costos estimado, así como un contexto de incertidumbre y contingencia implícitas al desarrollo de software.

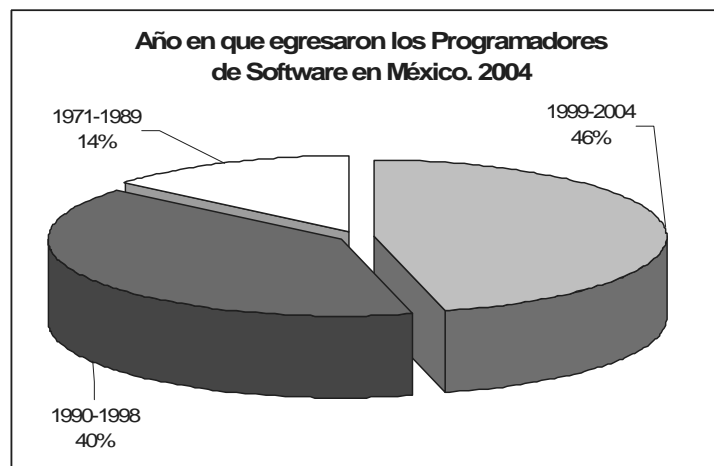
Con respecto a la persistencia del riesgo e incertidumbre en el desarrollo el software a pesar de la existencia de la Ingeniería del Software, se ha integrado una corriente de académicos y empresarios europeos (esto no quiere decir que no haya estadounidenses) que critican el método general de la Ingeniería del Software, en el sentido que la Ingeniería del Software se ha centrado casi exclusivamente en los atributos y particularidades del software, y ha descuidado al usuario final, se ha centrado más en temas relacionados con el sistema de operación interna, con el rendimiento y fiabilidad del sistema; es decir se ha centrado en aquellos factores que pueden ser medidos objetivamente como: numero de errores por cada numero de líneas de programación; tiempos de respuesta y probabilidades de error por cada determinado numero de código, etc.; se progreso en aspectos “duros”, deterministas, verticales que intentan solucionar la calidad del proceso de trabajo: Métodos y herramientas de análisis, diseño, codificación y pruebas; Revisiones técnicas formales; Estrategia de pruebas multi-escalada; Control de documentación de software y de los cambios realizados; Procedimientos de ajuste a los estándares de desarrollo de software; Mecanismos de medida y de información.

### **Una ocupación en construcción**

Los programadores de software son un conjunto de trabajadores recientes, podríamos señalar que esta en proceso la acumulación de conocimientos sociales de ésta profesión. Según datos de la grafica, 46% de los programadores culminaron sus estudios profesionales entre 1999 y 2004, y 40% entre 1990 y 1998, es decir 86% de los programadores poseen menos de 15 años de experiencia (tomando como referencia que la encuesta se realizo en 2004). Otra lectura significativa de la grafica 1 es aquella que muestra que sólo 14% de los programadores de Software posee una experiencia mayor a 15 años. Dato que a su vez es consistente con ANUIES (2007) que señala que los profesionistas empleados en México en el primer trimestre del 2007 en ingeniería en computación e informática, con rangos de edades de 20 a 34 años representan 73.2%, que culminaron sus estudios entre 1992 y 2005, porcentaje similar al de la grafica 1 que observa 86% de los profesionistas del software culminaron sus estudios profesionales entre 1990 y 2004.



**Grafica 1**



Fuente: elaboración propia en base a "Estudio para Determinar la Cantidad y Calidad de Recursos Humanos Necesarios para el Desarrollo de la Industria de Software en México" Secretaría de Economía-UAM. 2004.

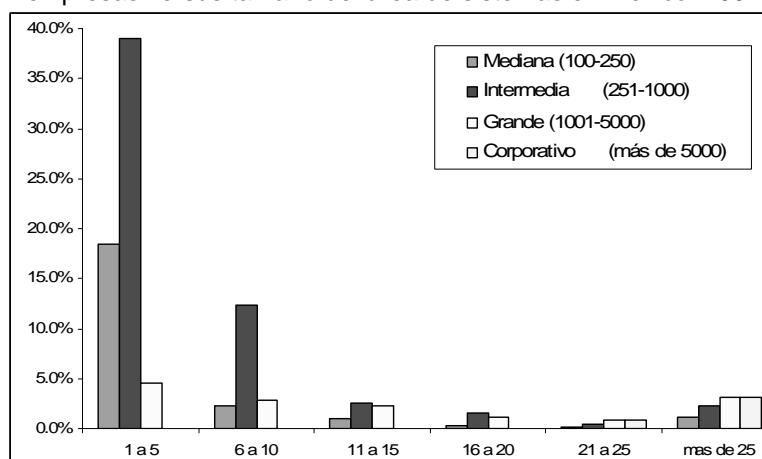
En la literatura de ingenieros es común percibir que la industria del software esta en proceso de consolidación, que hace falta superar la "etapa artesanal", concepto común entre gerentes y líderes de empresas de Software. La "etapa artesanal" hace referencia no sólo al hecho de que la ingeniería del software es relativamente nueva en comparación a "otras" ingenierías en donde las ocupaciones están socialmente divididas en arquitectos, ingenieros, ingeniero en electrónica; ingeniero químico, etc. También hace referencia al contexto "joven" de los programadores puesto que: 86% se graduaron después de 1990; 87% están centralizados en equipos de trabajo con menos de 10 empleados y, 83% están concentrados en micros, pequeñas y medianas empresas.

El proceso de trabajo del software mexicano, no sólo posee pocos años de madurez en conformar sus relaciones sociales de producción, comprende un trabajo de reciente formación, está concentrado en PyMES en más de 90% de los casos y centrado en pocos trabajadores (ver grafica 2). Esta concentración y lo reciente del trabajo, explica en parte porque en algunas entrevistas con gerentes de software señalaban que existe alta rotación laboral, insuficientes habilidades y conocimientos entre programadores "lo mas difícil es ponerme a enseñarles...a decirle que deben de empezar a hacer....pero aprenden rápido" (Entrevista 1 Líder).

Las empresas que poseen un centro de desarrollo de software, según corresponde a la grafica 2, 39% están concentradas en empresas medianas (100 a 250 empleados) y 17.4% en intermedias (251 a 1,000 empleados); ahora bien las áreas de desarrollo de software están centralizadas en unas cuantas personas, según vemos en las abscisas de la grafica 2 las empresas que poseen áreas de sistemas, están centralizadas en menos de 5 empleados, es decir 69.5% de los equipos de trabajo del área de sistemas se compone de 1 a 5 empleados, y 17.5% de 6 a 10 profesionistas ocupados en empresas que poseen sus propios centro de desarrollo de sistemas informáticos, concentrándose 87% de áreas de sistemas informáticos con menos de 10 trabajadores (ver grafica 2).

**Grafica 2**

Distribución de los profesionistas del Software por tamaño de empresas versus tamaño del área de sistemas en México. 2004



Fuente: elaboración propia en base a "Estudio para Determinar la Cantidad y Calidad de Recursos Humanos Necesarios para el Desarrollo de la Industria de Software en México" Secretaría de Economía-UAM. 2004.

El trabajo de software en las empresas desarrolladoras a su vez está concentrado por tamaño de empresa en micros, pequeñas y medianas empresas. En México estas empresas concentran 92% de empresas desarrolladoras. Según vemos los datos del cuadro 1, para septiembre de 2006 de 1,492 empresas que ofrecen algún tipo de desarrollo de Software, 41% son microempresas que contratan a menos de 10 trabajadores, porcentaje que se duplica (83%) cuando observamos en conjunto de las micros y medianas empresas.

**Cuadro 1**

Empresas de Software en México y en la zona metropolitana del Distrito Federal

	Rango de empleo	México * (09-2006)	ZMDF ** (02.2005)	México %	ZMDF %
<b>Tamaño</b>		<b>1,492</b>	<b>88</b>	<b>100%</b>	<b>100%</b>
Micro	1 a 10	619	33	41%	38%
Pequeña	11 a 50	629	41	42%	47%
Mediana	51 a 00	130	8	9%	9%

Grande	Mas de 100	114	6	8%	7%
--------	------------	-----	---	----	----

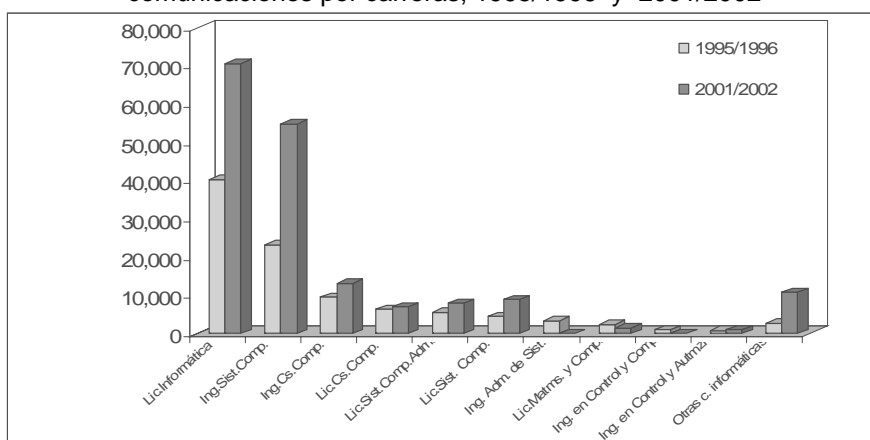
\*Datos de INEGI, Septiembre de 2006. Acceso a Software.com

\*\* Elaboración propia en base a datos de CANIETI, AMITI y AMCIS.

Otro ingrediente cualitativo que nos permite visualizar lo incipiente de la ocupación, es el abanico de ofertas entre ingenierías en computación y las licenciaturas en informática que son ofertadas por la educación superior (ANUIES-INEGI: 2006), según vemos en la grafica 18 tan solo la educación pública ofrece un abanico superior a 14 profesiones entre ingenierías en computación y licenciaturas en informática; a esta lista habría que agregar las de educación superior privada, diplomados y postgrados (públicos y privados).

**Grafica 3**

Matrícula de nivel licenciatura en tecnología de información y comunicaciones por carreras, 1995/1996 y 2001/2002



FUENTE: Elaboración propia en base a INEGI, <http://www.inegi.gob.mx> Acceso Julio de 2007

La concentración en la matrícula, según la grafica 18 a nivel licenciatura ofrecida por la educación superior se incremento significativamente en el ciclo escolar 2001/2002 y las preferencias académicas se orientaron a ingeniería en computación y licenciatura en informática. Según datos de la grafica para 2002, 40% de las inscripciones fueron en licenciatura en informática; 31% en ingeniería en sistemas computacionales y 7% en ingeniería en ciencias de la computación; éstas 3 profesiones concentran 78% de la demanda de las profesiones, tan solo las primeras dos carreras juntas, en 2002 se ubican en el cuarto lugar de las profesiones con mas población atendida y, ocupan el primer lugar en las ingenierías (ver grafica 3).

Un debate no escrito entre empleadores y empleados, es el perfil académico de quien se contrata: profesionista ó técnicos. Debate que lo percibimos en algunas entrevistas realizadas en la zona metropolitana en la ciudad de México, entre Mayo y Septiembre de

2006, observamos más de tres posturas entre líderes de proyecto y programadores, con respecto a que tipo de profesiones, calificaciones y cualificaciones deberían impartir las universidades y que perfil profesional debe contener a quien se contrata; por el momento, señalaremos tres posturas de los gerentes:

- Quienes señalan que la industria del software requiere de técnicos especializados, no profesionales que sólo desean “tirar código”, ignorando metodologías y métricas de trabajo.
- Quienes precisan que si bien es cierto, si deben ser profesionistas con carrera universitaria, éstos no desarrollan las cualidades que la industria requiere, es decir, no poseen el perfil profesional que la industria demanda.
- Otros, indican que no es suficiente la educación superior, deben existir vínculos con la industria local, es decir, no poseen práctica suficiente (desvinculación empresa-universidad) real de los conocimientos que demanda el mercado y las calificaciones de los profesionistas.

Estos señalamientos recurrentes entre gerentes, líderes de proyecto y programadores, no son contradictorios, por el contrario son expresión de la complejidad, ambigüedad y claros-oscuros de un proceso de trabajo de reciente presencia. Trabajo que se conforma a su vez de un complejo renglón tecnológico, que hace presuponer que no se contrata mano de obra descualificada, sino que está compuesto por mano de obra especializada. Estas posturas oscilaron entre los entrevistados que señalan la diferencia entre contratar técnicos profesionales y profesionistas universitarios. Aquellos que consideran que se debe contratar personal con perfil de técnicos, argumentan que es ello es deseable porque de ésta manera es en la empresa donde desarrollan las habilidades y conocimientos necesarios; en cambio cuando contratas un profesionista universitario éstos “aprenden, cuando mejoran sus capacidades personales...piden que les pagues mas o simplemente se van” (Gerente DC2-TADI, mayo de 2006). O bien, a medida que adquieren mayores conocimientos formales, acumulan experiencias, “saben hacer” mejor los procedimientos y, cuando eso sucede el programador se percibe así mismo que puede aspirar a mejores condiciones de trabajo, y emprende una postura de “hacen como que me pagan, hago como que trabajo”, otros asumen una posición de “primas donas” que lo único que quieren hacer es programar sin que nadie les moleste y sin involucrarse en otros temas como el diseño, pruebas de calidad, o en documentar el proceso (gerente RF2-DYYA, Junio de 2006). Por el contrario aquellos

gerentes y líderes que señalan que los programadores deben poseer preparación académica a nivel profesional, solo que debe haber cauces de comunicación empresas-universidad, impulsarse entre ambos espacios acuerdos para estancias, prácticas profesionales, e incluso que los empresarios formen parte de los académicos. Aproximadamente entre dos gerentes y cuatro líderes entrevistados coincidieron en utilizar el ejemplo del “eje tecnológico” (<http://www.tecnoeje.org/>). Este proyecto es financiado por fondos del programa de software (PROSOFT), Secretaría de Economía e involucra más de 15 universidades de la zona metropolitana del Distrito Federal, así como un programa de estancias y practicas profesionales, becas económicas, cursos y certificaciones, construcción de un edificio tecnológico para las prácticas profesionales, etc. (Gerente, JA2, MIXE, Mayo de 2006).

Está polémica entre contratar técnicos o profesionista en el proceso de trabajo del software, en un primera acercamiento parece no reflejarse en el mercado laboral, ya que cuando vemos los datos del cuadro 2 observamos que 78.3% de los ingenieros de software poseen una carrera profesional terminada y sólo 19.6% declaro ser técnico; 12.9%, declaro ser técnico medio superior y 6.7% superior universitario. Sin embargo, cuando observamos el detalle por grupos de edad, el debate se replantea, según podemos ver en el cuadro 2 que los profesionistas del software, no sólo esta conformado en su mayoría por “ingenieros o licenciados” como apreciamos al principio, sino que entre los mas jóvenes de 18-23 años 43.6% declaró ser técnico y 54.6% ser profesional, suponemos que este crecimiento en los jóvenes técnicos podrían corresponder a los jóvenes que se gradúan del sistema técnicos del país de reciente presencia en el sistema educativo mexicano, como son las Universidades tecnológicas.

## Cuadro 2

Formación académica versus edad de los profesionistas de Software en México.2004

	Total	18 - 23	24 - 30	31 - 35	36 - 40	+ de 40
Total	100.0%	8.9%	46.8%	21.7%	14.6%	8.0%
Técnico medio superior	<b>12.9%</b>	<b>33.7%</b>	8.7%	15.4%	8.1%	16.5%
Técnico superior universitario	6.7%	<b>9.9%</b>	6.4%	5.7%	6.1%	8.4%
Profesional	<b>69.3%</b>	<b>44.5%</b>	<b>75.5%</b>	<b>65.8%</b>	<b>71.4%</b>	<b>66.9%</b>
Profesional y Técnico	9.0%	10.0%	8.3%	9.5%	12.9%	4.1%
Ninguno	2.0%	2.0%	1.1%	3.6%	1.5%	4.1%

Fuente: elaboración propia en base a “Estudio para Determinar la Cantidad y Calidad de Recursos Humanos Necesarios para el Desarrollo de la Industria de Software en México” Secretaría de Economía-UAM. 2004

Otra lectura importante del cuadro 2, está relacionado con los rangos de edad, donde observamos que 68.5% son jóvenes entre 24 y 35 años, los cuales en su mayoría de cada diez, siete son profesionistas, por el contrario, los mas jóvenes de entre 18 y 23 años sólo cuatro de cada diez son profesionistas, dato que parecería ser contradictorio para una profesión que supone alta especialización. Este debate se complejiza cuando examinamos el conjunto de habilidades, conocimientos y destrezas que deben conocer los programadores de software, inmersos éstos en un contexto de reciente conformación, con trabajadores con pocos años de experiencia, donde: 76% posee grado universitario, 81% con menos de 15 años de experiencia y 77% son menores de 35 años.

Los expertos reconocen que el desarrollo de software como un proceso de trabajo inmaduro, cuasi-artesanal, de esta manera catedráticos como Pressman (2001) y Yurdón (2004) entre otros reconoce el carácter prematuro, no sólo de la industria, sino también de los profesionistas que la integran. En este sentido consideramos que la acumulación de conocimientos formales e informales, habilidades y destrezas del trabajador frente a la ocupación esta en un doble proceso de construcción: por un lado la construcción social de la profesión (mejora en los procesos tecnológicos, herramientas técnicas como nuevos lenguajes de programación, nuevos procedimientos) y por otro, el trabajador mismo esta inmerso en un proceso subjetivo de asentamiento, reconocimiento social de la profesión, legitimidad de esta frente al trabajo (nuevas formas de lucha por el poder y el control en el que hacer y como hacer).

Cuando los gerentes advierten que los programadores de Software deben ser mas “técnicos”, hacen referencia a la división social del trabajo hacia el interior de la empresa, es decir que los programadores se dediquen mas al aspecto práctico, cotidiano de la programación, de la codificación, proceso que no es sencillo, es complejo, álgido, ya que representa el punto en el cual se traduce el diseño del sistema informático en algoritmos que resuelve y soluciona los problemas planteados en el proceso de trabajo. Pero por otro lado, existe un problema central que es la complejidad en el proceso de abstracción, en el proceso de solución de problemas, que este trabajo implica para el programador. Complejidad que la ingeniería del software señala como “aflicción del software”, que representa un conjunto de problemas, errores y defectos en el proceso de trabajo, problemas como la falta de documentación y comentarios de el conjunto de abstracciones y soluciones algorítmicas,

incumplimiento en tiempos en la entrega, inobservancia en los requerimientos señalados por los clientes en el diseño original, etc.

Consideramos que las propias condiciones externas e internas en el proceso de trabajo, aunado a la interpretación subjetiva de los programadores llevan a que gerentes, y directivos y algunos académicos consideren que es pertinente impulsar el aspecto administrativo de la programación, que sea la gerencia quien realice el diseño, señale los requisitos de los módulos, y sólo encomendar a los programadores el aspecto de la codificación de los algoritmos. Lo que intentamos señalar es que, por un lado, se desea que el área administrativa predomine en el proceso de trabajo, ya que este es un trabajo altamente dependiente del trabajador-programador, no sólo incertidumbre en cuanto al procedimiento, sino también la documentación de lo que se hizo y porque se hizo tal procedimiento algorítmico.

“...yo no he tenido que darle cuentas a nadie, entonces, de ahí, me cuesta trabajo el especificar lo que tengo que hacer (documentar el proceso), por que muchas veces al no tener que rendirle cuentas a alguien, lo vas haciendo conforme ves que te van llegando a veces las ideas, esto te hace como hacer lo que vas hacer... pero si me preguntas ¿que voy hacer mañana?, igual todavía no lo defino...” (Programador, JO3-MIXE, Julio de 2006).

Parte de la aflicción del software, reside en el hecho que los programadores no sólo no documentan el proceso, también hay una alta incertidumbre en cuanto a porque hicieron determinado algoritmo. Sin profundizar en el concepto consideramos que esta presente una resistencia por el control del trabajo al interior del proceso de trabajo del programador. Lucha que representa una serie de consensos y arreglos sociales al interior del proceso de trabajo.

## **Conclusión**

La característica principal del trabajo cognitivo del software supone una redefinición de la actividad laboral a nivel de interacción cotidiana entre los actores que participan, como el programador, el cliente y el gerente, interacción que no necesariamente sucede en un contexto fordista, supeditada a ordenes del capataz o cumplir tiempos y movimientos de las maquinas; por tanto cabría señalar una construcción social de trabajo ampliado que se sucede al interior de comunidades simbólicas de trabajo (De la Garza, 2006a:16). Presuponemos que en *las comunidades cognitivas del trabajo de Software*, implican una

forma de organización laboral no Taylorista. Donde las relaciones laborales se configuran una serie de contingencias, opuestas al proceso de la manufactura. No nos referimos sólo a la participación del cliente, sino también a la yuxtaposición de las fases del proceso de producción y la objetivación del bien desarrollado fuera del consumidor y del proceso mismo de producción, así como la acumulación del saber hacer (información y conocimiento) entre las comunidades cognitivas.

*Las comunidades cognitivas de trabajadores de software a la medida* es un concepto mucho más amplio que busca explicar la *constelación de configuraciones subjetivas* que se suceden en la interacción dentro y fuera del proceso de trabajo y están comprendidas por un conjunto de trabajadores que establecen relaciones sociales amplias, extendidas hacia el interior como al exterior del proceso de trabajo en tiempo real y virtual, sincrónico y diacrónico; no son estructurales, verticales o rígidas, con interacciones que pueden ser cara a cara, pantalla a pantalla, en tiempo real o virtual, etc., no están limitadas o restringidas al ámbito laboral; es una comunidad con estructuras que le constriñen y con grados de libertad en la toma de decisiones que implica operaciones subjetivas a nivel individual o colectiva. *Las comunidades cognitivas de software*, estarían entonces constreñidas por una serie de estructuras que forman parte del proceso de trabajo: Existencia de varios tipos de software: software empotrado, software empaquetado, software a la medida, donde las características en el desarrollo del trabajo es variado. Pero, se mantiene la libertad del programador en la toma de decisión en la configuración de las líneas de código que dan origen a los algoritmos. Los algoritmos, representan un conjunto de *signos* y caracteres que tienen su origen en los lenguajes de programación (existen mas de 2000 leguajes). Los algoritmos es el aspecto lógico formal y están embebidos por diferenciados grados de incertidumbre (dependerá de la complejidad del algoritmo desarrollado). La incertidumbre de un algoritmo esta en relación directa con condiciones objetivas y subjetivas. Las objetivas son el grado de conocimiento del lenguaje de programación utilizado, experiencia medida en cantidad de programas desarrolladas en el lenguaje que se utiliza; comprensión metódica del problema planteado y “traducción” correcta de la instrucción en un algoritmo. Las configuraciones subjetivas, es la habilidad y destreza cognitiva en la resolución de problemas con el lenguaje de programación que se pretende utilizar. En otras palabras es el procedimiento reflexivo para escribir el requerimiento en un código lógico y coherente que hemos denominado como texto sígnico.



En el proceso de trabajo del software no aplican los procedimientos clásicos del trabajo, es en referencia al hecho que el software no está restringido a las “leyes” de la manufactura tradicional o por las límites físicos de la fábrica e incluso de ritmos impuestos por la gerencia propios de la producción en serie del fordismo y el toyotismo. El proceso de trabajo del software no es material es cognitivo, el trabajo no es de esfuerzo físico, es reflexivo; la eficiencia y la productividad del trabajo no se ajusta a las economías de escala y aunque las Ingenierías del Software proponen medir la productividad y eficiencia con métricas como son la producción por hora hombre o errores por cada ciento de programas desarrollados, o implementar estándares de calidad como CMM; ISO, IEE, etc., éstas no resultan las mas adecuadas por caras, burocráticas y rígidas. Las economías posproducción de la manufactura no aplican en el desarrollo de Software porque el primer software concentra el costo total invertido y el segundo software clonado, su costo tiende a ser equivalente al costo del dispositivo donde se genere la copia. Es decir en el trabajo clásico, los costos forman parte importante de la post-producción, en cambio en el trabajo del software el costo se concentra en el primer producto. Sin embargo, si se presentan errores en el software post-producción, no significa que sean errores generados en el uso, sino que ya estaban ocultos en el sistema, es decir *no hay software con cero errores*, como pretende la fabricación flexible; el software contiene fallas implícitas al *ciclo de desarrollo* (periodo de tiempo del proceso de trabajo que tarda en desarrollar el software) que se forman en el diseño (conceptualización y formalización de los requerimientos) o bien en el procesamiento de los datos (generación de rutinas de algoritmos o bucles de códigos) y se exteriorizan en las pruebas de calidad o en la implementación (con el cliente o el usuario final), es decir *no hay software con cero defectos*. Es importante señalar que el software con errores y defectos no se rechaza, no se destruye (en el mejor de los casos, obviamente dependerá de la magnitud de los mismos), se “parcha” y se asume como inevitable que se incrusten nuevos dispositivos de “actualización”, sin embargo el error y defecto que se manifiesten con el uso, no son nuevos, ya estaban presentes en el producto y afectan a todas las copias del software. La falta de métricas y metodologías de calidad eficientes que comprueben no sólo los errores y defectos, sino que evalúe el progreso, el cumplimiento de tiempos y proceso implica que constantemente se sobrepasen los tiempos acordados para la entrega del proyecto, es decir *no hay software justo a tiempo*. Aún con *herramientas procedimentales* de cuarta o quinta generación (programación orientada a objetos y aspectos, métricas de calidad tipo CMM, estándares IEEE, etc.) el proceso de trabajo contiene un alto contenido de “*creado a mano*”

en el software a la medida. Este conjunto de inconsistencias; errores, fallos, defectos no se termina a tiempo, etc. están contenido en una serie de contingencias denominada “aflicción del software”. Paralelamente a este conjunto de aflicciones no hay una planificación sistemática del proceso de trabajo; escasa aplicación de normas de calidad; insuficiencia o limitación en el seguimiento y gestión del proyecto desarrollados; necesidad de personal con conocimientos, habilidades, destrezas, experiencias en nuevos métodos y plataformas tecnológicas. Así como un entorno polarizado en las calificaciones impartidas por el Estado (heterogeneidad en las profesiones) y en las estrategias que implementan las empresas a nivel individual (polarización industrial en el software a la medida).

El proceso de trabajo del software es atravesado transversalmente por una ausencia de recetas únicas, además de no existir un control total de calidad, cero errores, cero defectos e incumplimiento en un justo a tiempo, se suma la imprevisibilidad de no existir “un solo mejor camino” para desarrollar los proyectos de software. En las entrevistas los líderes y gerentes sostenían que el proceso de trabajo en el software, debería ser como la manufactura de automóviles, donde existían tolerancias al error, rangos para las fallas por cada número de unidad producidas, etc., igual correspondería al software. Sin embargo, se contestaban así mismos, cuando reconocían que cada proyecto de software es *sui generis*, con contextos determinados, únicos e impredecibles; cada proceso de trabajo para desarrollar un software posee lógicas propias, internas, que varían entre proyectos. Cada software se enfrenta a contenidos diversos, (requerimientos a la medida), necesidades de actualización y mantenimientos específicos, es decir, que los programas se modifican a lo largo del ciclo de vida.

## **Bibliografía**

1. Arora, A., A., Fosfuri, y A., Gambardela (2002), Los mercados de tecnologías en la economía del conocimiento, en Foray D., Sociedad del conocimiento, Revista internacional de ciencias sociales, numero 171, Marzo. Pp. 155-174.
2. Casas, R., (2003), Enfoque para el análisis de redes y flujos de conocimiento, Luna, M., (Coord.) Itinerarios del conocimiento: formas dinámicas y contenido. Un enfoque de redes, Ed. España (Anthropos) México (Universidad Autónoma Metropolitana Izt.) pp. 19-50.
3. Chudnovsky, D., López, A. y Melitsko, S. El sector de software y servicios informáticos en la Argentina: Situación actual y perspectivas de desarrollo CENIT DT 27/07 2001.

4. Correa, C. (1990), "The legal protection of software. Implications for latecomer strategies in newly industrializing economies and middle-income economies", OECD Development Center, Technical Paper N° 26, Paris.
5. D'Costa, A. P. (2000), "Export Growth and Path-Dependence The Locking-in of Innovations in the Software Industry", 4th International Conference on Technology Policy and Innovation, Curitiba, Agosto.
6. David, P., y Foray D., (2002) Una introducción a la economía y a la sociedad del saber, en Foray D., Sociedad del conocimiento, Revista internacional de ciencias sociales, numero 171, Marzo. Pp. 7-28.
7. De la Garza E, (2001), La formación socioeconómica neoliberal. México, Plaza y Valdez
8. De la Garza E., (1996), Modelos de industrialización en México, DF., UAM.
9. De la Garza Enrique, (Coord.), (2003), Tratado Latino Americano de Sociología del Trabajo, Colegio de México, FLACSO; UAM, Siglo XXI. México.
10. De la Garza, Enrique (1994). "Neoliberalismo y Estado" en Asa C Laurell, Estado y Políticas Sociales en el Neoliberalismo. UAM –X. Distrito Federal. PP. 59-73
11. Hardt. M. y A. Negri (2001). Empire, Cambridge: Harvard University Press.
12. Pérez, C., (2003) Revoluciones tecnológicas, cambios de paradigma y de marco socioinstitucionales, en Aboites, J., y Dutréint, G., (Coords.) Innovación, aprendizaje y creación de capacidades tecnológicas, UAM, Xochimilco. Pp. 13-45
13. Torrisi, S. (1998), Industrial Organization and Innovation. An International Study of the Software Industry, Edward Elgar, Cheltenham.
14. Yoguel, G. y Boscherini, F., (2001), "El desarrollo de las capacidades innovativas de las firmas y el rol del sistema territorial.", en Revista Desarrollo Económico.
15. Yoguel, G.; Borello, J.; Erbes, A.; Robert, V.; Roitter, S. (2004). Competencias tecnológicas de los trabajadores informáticos argentinos. Más allá de las restricciones de demanda y oferta. Littec e-papers.
16. Godolier Maurice (1989), Lo ideal y lo material, Madrid, Taurus Humanidades. Gintis. Herbert (1983) "La naturaleza del intercambio laboral y la teoría de la producción capitalista; en Tohara, L. (Comp.) El mercado de trabajo: teorías y aplicaciones, Madrid, Alianza Universidad, 1986, p. 176.

\* Jose G. Rodríguez-Gutiérrez, Dirección: Andador Huatabampo 6-A y Calle Yecora, Colonia: FOVISSSTE, Hermosillo, Sonora, México. C.p. 83020. Tel.part. 01662-2541720; [joserodriguez@nogales.uson.mx](mailto:joserodriguez@nogales.uson.mx); Profesor en la Universidad de Sonora, Unidad Nogales. Estudiante de doctorado en la Universidad Autónoma Metropolitana, Iztapalapa. Temas de Investigación: organización del trabajo en la Industria Maquiladora y Servicios. Tesis Doctoral: Control sobre el proceso de trabajo entre trabajadores cognitivo de la Industria del Software en el valle de México.