Revista Antioqueña de las
Ciencias Computacionales y la Ingeniería de Software

**ISSN: 2248-7441**

*www.fundacioniai.org/raccis*
*raccis@fundacioniai.org*

# Manifesto for the Software Development Professionalization

# Manifiesto por la Profesionalización del Desarrollo de Software

**Red Latinoamericana en Ingeniería de Software (RedLatinaIS)**
*http://fundacioniai.org/red.html. redlatinais(AT)lacrest.org*

**INFORMACIÓN DEL ARTÍCULO**

*Tipo de artículo*: Reflexión

*Historia del artículo*
Recibido: 20-11-2013
Correcciones:
Aceptado: 22-12-2013

*Categories and Subject Descriptors*
K.7.3 [**Computing Milieux**]: THE COMPUTING PROFESSION - *Methodologies*. Testing, Certification, and Licensing

*General Terms*
Software Testing, Software Engineering, Requirements Engineering.

*Keywords*
Requirements Engineering, Security, Reliability, Software Testing.

*Palabras clave*
Ingeniería de Requisitos, Seguridad, Fiabilidad, Prueba del Software.

**ABSTRACT**

One of the central problems of current economic development and industrial competitiveness, social and scientific, is the complexity of large and intensive software systems, and processes for their development and implementation. This complexity is defined by the amount and heterogeneity of the interaction of the hardware with the software components, their inter-relationships, of incorporation of the technical and organizational environments, and the interfaces to humans. The domain of these systems requires actions and scientific thoughts, hierarchical and systematic; also, the success of the products, services and organizations, is increasingly determined by the availability of suitable software products. Therefore, highly qualified professionals, able to understand and master the systems, involved in the entire life cycle of software engineering, and adopt different roles during the development. This is the reason that guide the thinking of this Manifesto , which aims is to achieve the *Professionalization of Software Development*.

**RESUMEN**

Uno de los problemas centrales del desarrollo económico actual y de la competitividad industrial, social y científica, es la *complejidad* de los grandes e intensivos sistemas software, y de los procesos para su desarrollo y aplicación. Esta complejidad se define por la cantidad y heterogeneidad de la interacción del hardware con los componentes software, de sus inter-relaciones, de la incorporación en los entornos técnicos y organizacionales, y de las interfaces para los seres humanos. El dominio de estos sistemas requiere de acciones y pensamientos científicos, jerárquicos y sistemáticos; además, el éxito de los productos, los servicios y las organizaciones, está cada vez más determinado por la disponibilidad de productos software adecuados. Por lo tanto, se necesitan profesionales altamente cualificados, capaces de comprender y dominar los sistemas, de participar en todo el ciclo de vida de la Ingeniería de Software, y de adoptar diferentes roles durante el desarrollo. Esta es la razón que guía el pensamiento de este Manifiesto, cuyo objetivo es lograr la *Profesionalización del Desarrollo de Software*.



## 1. INTRODUCCIÓN

El explosivo crecimiento de las Tecnologías de la Información ha generado desafíos particulares para el conocimiento, en relación con el uso de los términos *ingeniero de software* e *Ingeniería de Software*. Actualmente, muchos de quienes desarrollan software se refieren a sí mismos como *ingenieros de software*, y a su trabajo como *Ingeniería de Software*, a pesar de no contar con una formación en ingeniería y de no estar reglamentados de alguna forma. Dadas la penetración e importancia que el software ha alcanzado en este siglo esta situación se ha convertido en un problema, porque induce a errores en los productos y una interpretación y valoración inadecuada de la profesión. Además, la dependencia social del software ha llegado a niveles amplios, y las personas delegan proyectos software a estos *seudo-profesionales*. Debido a que construir software se ha definido popularmente como una cuestión informal, la mayoría de ellos se sienten calificados para desarrollar programas, sólo por el hecho de conocer la semántica y la gramática de algún lenguaje. Esta cuestión ha hecho carrera, al punto que la industria se ha dejado influenciar y los contrata para *desarrollar* los sistemas, que luego ofrece al público. La práctica de esta ingeniería debe estar autorizada y regulada, casi al mismo nivel que la medicina o la ingeniería civil, porque si el desarrollo de Sistemas de Información no cumple con los criterios necesarios de integridad, eficiencia, fiabilidad y correctitud se podría poner en riesgo la vida de las personas y los intereses económicos de las empresas.

Por otro lado, la Ingeniería de Software proporciona el marco teórico, los métodos y las herramientas necesarios para desarrollar software de calidad, y ha impulsado la revolución de la Sociedad de la Información y del Conocimiento, porque sin sus aportes los computadores

serían sólo herramientas sin utilidad específica. Además, a pesar de los avances en *hardware*, el impacto y la potencialización del desarrollo tecnológico sólo fue posible gracias a los productos *software*. Por otro lado, la actual sociedad se empieza a reconocer como *software-dependiente*, porque en este siglo el software hace parte de todos los dispositivos que se requieren para manipular información, y que las personas utilizan en sus actividades cotidianas.

El término *Ingeniería de Software* se conoció por primera vez en la NATO Software Engineering Conference de 1968, e inicialmente su objetivo era concientizar acerca de la *crisis del software* que se vivía en ese momento. Desde entonces, se ha desarrollado como una profesión y como un campo de estudio, cuyo objetivo es desarrollar productos de mayor calidad, más asequibles, fáciles de mantener y rápidos de construir. Dado que es un área del conocimiento relativamente joven, en comparación con sus campos ingenieriles afines, todavía se presenta debate en torno a lo que realmente es y si se ajusta a la definición clásica de ingeniería. Por ejemplo, las prácticas necesarias para crear un buen producto software se establecieron hace décadas, pero a pesar de algunos triunfos sobresalientes esta industria todavía no está a la altura de sus capacidades. Existe un amplio abismo entre las necesidades reales del medio y los productos desarrollados, e inclusive muchas de las prácticas de uso generalizado son anticuadas. Algunos profesionales utilizan métodos de desarrollo de software tratando de imitar los procesos que aplica la ingeniería civil, y otros han perdido la paciencia con antiguos modelos burocráticos, de cascada y con documentación abundante, que aunque útiles en su momento actualmente se olvidaron de, o apenas tiene en cuenta, conceptos fundamentales como calidad, accesibilidad, facilidad de mantenimiento, agilidad, criterio y orientación a resultados, que son características deseables en todo tipo de producto software para este siglo.

Actualmente, uno de los enfoques más comunes para desarrollar software en el mundo es la programación mediante codificación y corrección, en el que un equipo de programadores inicia con una idea general de lo que quieren construir —puede que tengan una especificación formal, pero probablemente no—, y utilizan la combinación informal de diseño, codificación, depuración y metodologías de prueba que más les convengan, para luego escribir un poco de código y ejecutarlo para ver si funciona, y, si no funciona, lo modifican hasta lograr el objetivo. Pero este enfoque no concuerda con el actual estado del arte en desarrollo de software, porque en el proceso se evidencia una Ingeniería de Requisitos inadecuada, lo que genera productos de inferior calidad que los otros métodos, aunque su principal ventaja es que requiere poca formación técnica o de gestión. Desde hace décadas, las organizaciones líderes en el área conocen y utilizan prácticas eficaces de desarrollo de software, pero la brecha entre la práctica promedio y las mejores prácticas en software es amplia.
En este sentido, si uno de los extremos del espectro de competencias del desarrollo de software lo ocupan la codificación y la corrección, el otro lo está por la Ingeniería de Software, es decir, un enfoque sistemático, sistémico, disciplinado y cuantificable, aplicado a la concepción, desarrollo, operación y mantenimiento de software. La comunidad, que trabaja en este sentido, ha sido testigo de acontecimientos alentadores relacionados con el establecimiento, soporte y difusión de amplios estándares de gestión para la práctica del desarrollo de software, y este documento es un aporte más a esos sucesos, con el objetivo de documentar, proponer y apoyar el movimiento por la *Profesionalización del Desarrollo de Software*.

Además, un problema central del desarrollo económico actual y de la competitividad industrial, social y científica, es la *complejidad* de los grandes e intensivos sistemas software, y de los procesos para su desarrollo y aplicación. Esta complejidad se define por la cantidad y heterogeneidad de la interacción del hardware con los componentes software, de sus inter-relaciones, de la incorporación en los entornos técnicos y organizacionales, y de las interfaces para los seres humanos. El dominio de estos sistemas requiere de acciones y pensamientos jerárquicos y sistemáticos, y el éxito de los productos, los servicios y las organizaciones, está cada vez más determinado por la disponibilidad de productos software adecuados. Por lo tanto, se necesitan profesionales altamente cualificados, capaces de comprender y dominar los sistemas, de participar en todo el ciclo de vida de la Ingeniería de Software, y de adoptar diferentes roles durante el desarrollo Esta es la razón que guía el pensamiento de esta Manifiesto, cuyo objetivo es lograr la *Profesionalización del Desarrollo de Software*.

Este documento contiene una recopilación de ideas y aportes de diferentes actores de la comunidad latinoamericana del software, y tiene como objetivo constituir la base conceptual que sustente los principios de la Red Latinoamericana en Ingeniería de Software (RedLatinaIS). Este manifiesto no es el primer trabajo de este tipo, y esperamos que no sea el último, pero estamos convencidos que será una contribución importante para el logro de las iniciativas de la Red. Están todos cordialmente invitados a colaborar con sus aportes e ideas para lograr el establecimiento de un punto común de entendimiento alrededor de la necesidad de profesionalizar el desarrollo de software.

## 2. DEFINICIONES Y PRECISIONES NECESARIAS

Debido a la amplia variedad de conceptos y términos que se involucran en el contexto en el que se suscribe el *Manifiesto*, a continuación presentamos algunas definiciones, construidas desde nuestra experiencia y conocimiento, para dar claridad a su intencionalidad en el artículo.

### 2.1 Sistema

En su definición más amplia, un sistema es un conjunto integrado de elementos que cumplen un objetivo determinado, y desde diferentes disciplinas de la ingeniería se tienen diferentes perspectivas de lo que es. Por ejemplo, para la Ingeniería de Software es un

conjunto integrado de programas informáticos, y para la Eléctrica se refiere a los circuitos integrados, o a un conjunto de unidades eléctricas, es decir, su significado depende de la perspectiva y el contexto en el que se aplique. En su definición más interna, es una conjunción de recursos y procesos que operan para lograr un propósito común, y por lo tanto satisfacer alguna necesidad. Es decir, es un conjunto de componentes interactivos o interdependientes que forman un todo integrado.

### 2.2 Software

En términos generales se refiere a la información que se almacena en algún tipo de medio digital. El software es la interfaz y la lógica abstracta que vincula al ser humano con la tecnología tangible y otro software, permitiendo una mutua interacción y retroalimentación. En una visión más técnica, el término se utiliza para describir una colección de programas informáticos, algoritmos, procedimientos y la documentación necesaria que realizan algunas tareas en un sistema, como operar los equipos y dispositivos tecnológicos relacionados, y la gestión de la información para la toma de decisiones.

### 2.3 Ciencias Computacionales

Las Ciencias Computacionales no se refieren a la construcción de computadores o a la escritura de programas, ni a las herramientas utilizadas en la computación, se refieren a cómo utilizar estas herramientas y a interpretar lo que se encuentra en el proceso. La solución a muchos de los problemas de estas ciencias ni siquiera requiere el uso de computadores, sólo lápiz y papel, e incluso la mayoría se ha abordado y solucionado desde décadas antes que se construyeran las máquinas modernas. Estas ciencias se refieren al enfoque científico y matemático de las tecnologías de la información y sus aplicaciones, y al software y al hardware subyacente. Un científico computacional se puede especializar en la teoría computacional, o en la experimentación y diseño de componentes y nuevas tecnologías. Abarcan un amplio rango de conocimientos, desde los fundamentos teóricos y algorítmicos hasta los desarrollos de vanguardia en robótica, visión por computador, sistemas inteligentes, bioinformática, y otras áreas relacionadas. Los científicos computacionales pueden diseñar e implementar software, crear nuevas formas de utilizar los computadores y desarrollar métodos eficaces para resolver problemas computacionales, y en ocasiones, los planes de estudios de área de formación son criticados por no preparar a los estudiantes para un trabajo específico, porque mientras otras disciplinas lo hacen en habilidades para el trabajo inmediato, éstos ofrecen una formación integral de adaptación a nuevas tecnologías.

En términos generales, estas ciencias se ocupan del estudio sistemático de la viabilidad, la estructura, la expresión y la mecanización de los procesos metódicos o algoritmos, que subyacen a la adquisición, representación, procesamiento, almacenamiento, comunicación y acceso a la información, cuando está codificada en bits y bytes en un computador, o transcrita en los genes y en las estructuras proteínicas en una célula humana. La cuestión fundamental subyacente es averiguar qué procesos computacionales se pueden automatizar e implementar eficientemente. Para responder esta pregunta, aparentemente simple, los científicos computacionales trabajan en diversas áreas complementarias, como la naturaleza misma de la computación, para establecer cuáles problemas son o no computables; comparan diversos algoritmos, para determinar si ofrecen una solución correcta y eficiente a un problema; diseñan lenguajes de programación, para especificar y expresar algoritmos; diseñan, evalúan y construyen sistemas que ejecuten eficientemente esas especificaciones, y aplican los algoritmos al dominio de aplicaciones importantes.

### 2.4 Ingeniería de Sistemas

Es un área del conocimiento que en diversos países latinoamericanos no tiene una definición de consenso, aunque tradicionalmente se define como un campo multidisciplinar para construir grandes cosas complejas, y para lograrlo aplican métodos ingenieriles. Es un campo inter y multi-disciplinar de la ingeniería, que se centra en cómo diseñar y gestionar los ciclos de vida de los proyectos ingenieriles, y se ocupa de los procesos de trabajo y de las herramientas para gestionar los riesgos en este tipo de proyectos. Muchas veces se confunde con disciplinas técnicas y centradas en lo humano, como la ingeniería de control, la ingeniería industrial, los estudios organizacionales y la gestión de proyectos. Cuestiones como la logística, la coordinación de los diferentes equipos y el control automático de las máquinas se hacen más difíciles cuando se trata de proyectos grandes y complejos, pero para lograrlo, esta ingeniería integra varias disciplinas y grupos de especialistas en un esfuerzo de equipo para conformar un proceso de desarrollo estructurado, que va desde el concepto de producción hasta el de operación. Por lo sus practicantes deben considerar al negocio y a las necesidades técnicas de todos los clientes, con el objetivo de ofrecer un producto de calidad que satisfaga las necesidades generales.

La Ingeniería de Sistemas es un área que se ha desarrollado desde hace relativamente poco tiempo, que aplica técnicas y métodos de la Teoría General de Sistemas y de la matemática aplicada para dar solución a problemas complejos en diferentes campos del conocimiento. Su principal trabajo es asumir a los sistemas de ingeniería como conjunto y no como un componente en particular. Su objetivo surge del hecho de que casi todo lo que nos rodea y que utilizamos a diario es un sistema relativamente complejo, conformado por una serie de diferentes componentes mecánicos, electrónicos, hardware, software, y posiblemente otros tantos. Abarca el diseño de modelos matemáticos y el análisis de sistemas, centrándose especialmente en cómo encajan entre sí los distintos componentes, y en garantizar un diseño en el que todos interactúen de manera eficiente y eficaz. Muchas universidades de la región orientan este objeto de formación al desarrollo de proyectos software, aunque con sus planes curriculares no lo logren. Por otro lado, la industria tampoco tiene claridad acerca del perfil de estos ingenieros.

### 2.5 Ingeniería Informática

Es un campo del conocimiento que tiene que ver con el diseño y la construcción de equipos y sistemas basados en computadores. Se trata del estudio de hardware, software, comunicaciones, y la interacción entre ellos. Sus planes de estudio se centran en las teorías, los principios y las prácticas de la ingeniería eléctrica tradicional y de las matemáticas, que luego aplica para resolver los problemas de diseño de computadores y de dispositivos basados en ellos. En esta ingeniería se estudia el diseño de sistemas hardware digitales, incluyendo los de comunicaciones, los computadores y los dispositivos que contienen procesadores, y especialmente los dispositivos digitales y sus interfaces con los usuarios y otros dispositivos.

Los ingenieros informáticos analizan y diseñan el hardware, el software y los sistemas operativos para los sistemas informáticos. Para esto deben combinar los campos de las Ciencias Computacionales y la Ingeniería Eléctrica, por lo que a menudo se confunde con las primeras; pero los ingenieros informáticos también se forman en el diseño de software y en su integración con el hardware, y deben aplicar algoritmos y principios de diseño digital para diseñar, construir y probar los componentes utilizados para procesar, comunicar y almacenar la información, que normalmente se integra en grandes sistemas de ingeniería y en ambientes de redes distribuidas. Sus áreas de aplicación incluyen a las comunicaciones, la automatización, la robótica, la potencia, la energía, la salud, los negocios, la seguridad, el entretenimiento, y muchos otras.

### 2.6 Ingeniería de Software

Es la disciplina ingenieril que proporciona y aplica los métodos y herramientas necesarios para construir software de calidad, ajustado al presupuesto, en un plazo determinado y en un contexto de cambio constante de requisitos. Es la aplicación de un enfoque sistemático, disciplinado y cuantificable, al desarrollo, operación y mantenimiento de software; es decir, es la *aplicación de ingeniería al software*. En términos de costo y complejidad, en la mayoría de sistemas el software es el componente predominante, por tanto, las buenas prácticas de esta ingeniería y las herramientas pueden hacer una diferencia sustancial, incluso en la medida en que son la fuerza impulsora del éxito del proyecto. Se diferencia de las Ciencias Computacionales en que éstas tienen que ver con el desarrollo de aplicaciones científicas a gran escala, mientras que la Ingeniería de Software abarca no sólo los aspectos técnicos del desarrollo, sino también las cuestiones de gestión, como la dirección de equipos de diseño, de desarrollo, de pruebas, de presupuestos y de mantenimiento.

Actualmente, esta ingeniería ha evolucionado en respuesta a los factores del creciente impacto y costo de los grandes sistemas software en una amplia gama de situaciones, y a la importancia cada vez mayor del software en aplicaciones de seguridad crítica. Tiene un carácter diferente al de otras disciplinas ingenieriles debido a la naturaleza intangible de sus productos y a su discontinuidad operativa. Es un área del conocimiento que integra principios matemáticos y de las Ciencias Computacionales en las prácticas ingenieriles, para desarrollar el software que requieren los artefactos tangibles y físicos que utiliza la sociedad. En términos filosóficos, la Ingeniería de Software se encarga de la aplicación de diferentes estrategias y disciplinas para mejorar la capacidad de los seres humanos para enfrentar los retos de la actual Sociedad de la Información y el Conocimiento.

### 2.7 Programador

Los programadores escriben *buen código*. Hacerlo bien y limpio es un factor importante, pero a menudo tienen prioridad otras cuestiones. Las habilidades matemáticas son opcionales, aunque poseerlas les ayuda a ser conscientes de los problemas comunes y de las soluciones relacionadas con el dominio, pero son primordiales las relacionadas con la comunicación y la interacción social. Son generalistas sin un tipo de especialización verdaderamente profundo; son capaces de encontrar caminos en torno a los problemas, y de conectar diversos componentes para cumplir con una serie de requisitos. Su trabajo consiste en aplicar el conocimiento que poseen, del lenguaje de programación, para escribir código *eficiente* y *eficaz*, a la vez que respetan los conceptos de calidad y de seguridad, y responden al diseño que se ha construido desde la Ingeniería de Software.

### 2.8 Desarrollador

Los desarrolladores escriben *código de calidad*. Hacerlo limpio, claro, bien factorizado y libre de errores son cuestiones importantes que tienen en cuenta, y además conocen el significado de *buen código* dentro de un dominio. Tienen adecuados conocimientos en matemáticas y en cómo buscar buenas soluciones para los problemas; poseen amplios conocimientos en algoritmia y buenas habilidades en su área de experticia y las relacionadas; debido a que su trabajo se desenvuelve al interior de equipos multi-disciplinares, poseen buena capacidad de comunicación verbal y escrita, y de interacción con otras personas. Son profesionales que contribuyen de diversas maneras para que el producto software tenga éxito, y su trabajo consiste en aplicar principios científicos e ingenieriles para comprender, abstraer y moldear un problema, que se pueda resolver mediante un programa informático, para luego aplicar una metodología con el objetivo de llevar a la práctica la solución creada, tradicionalmente soportada en código de lenguaje de programación. En términos generales son responsables de aplicar procesos de calidad en todo el ciclo de vida del software.

## 3. LA INGENIERÍA DE SOFTWARE COMO PROFESIÓN

Por décadas, los investigadores han tratado de encontrar la *bala de plata* necesaria para matar al monstruo de la generación de software ineficaz. Estos esfuerzos se reflejan en la continua búsqueda de metodologías, lenguajes, notaciones y otras fórmulas, casi *alquimistas*, cada una comprometida en resolver un problema, y aunque muchas de ellas han hecho aportes sustanciales

no han logrado descubrir la bala faltante. Desde hace algún tiempo se inició un movimiento a nivel mundial que busca no encontrar una solución alquimista sino profesionalizar la labor de la Ingeniería de Software: el *Desarrollo de Software*.

La forma tradicional y ampliamente extendida de desarrollar software en diferentes industrias no se puede considerar una actividad profesional, porque en la mayoría de casos sus practicantes no aplican un modelo estructurado en el que deban emplear los niveles de destreza y las habilidades necesarios para lograr un producto con los niveles de calidad que la sociedad requiere. Este es el caso del software comercial, como los sistemas operativos y los ofimáticos, que al poco tiempo de salir al mercado deben ofrecer actualizaciones para corregir errores que se pudieron evitar desde el desarrollo. A diferencia de ingenierías como la Civil, en la que están claramente establecidos los requisitos formales para que sus practicantes se puedan considerar profesionales, la Ingeniería de Software parece no tener establecidas las reglas para que su ejercicio se considere profesional. Aunque algunas personas tienen instinto natural para *programar*, sin haber recibido capacitación formal para hacerlo, esto no quiere decir que tengan la formación necesaria para entregar productos fiables a la sociedad, sobre todo cuando se trata de sistemas críticos, como en las áreas aeroespacial, de la salud y la aviación. Tradicionalmente se acepta que un programador es bueno porque entrega el código esperado, pero hacen falta estándares de clasificación, aceptados de forma generalizada en la industria, la academia y el Estado, acerca de qué debería ser un *desarrollador profesional*, y de las características que se debe exigir para considerarlo como tal.

Para ilustrar esta cuestión basta con leer la declaración de la misión de Visual Basic para democratizar la *programación*: cualquiera que pueda arrastrar y soltar controles y comprender un mínimo de material técnico podría *juntar* una solución a un problema con un esfuerzo relativamente pequeño. Además, existen herramientas y servicios públicos que les permiten a los usuarios crear sofisticadas hojas de cálculo sin poseer conocimientos de programación práctica. Diversas personas e industrias han decidido apoyar iniciativas como las de la *Programación Ágil*, sin tener un conocimiento adecuado de ellas y de sus principios, sólo con el objetivo de encadenar servicios en línea sin los principios necesarios para un desarrollo de software fiable y de alta calidad. Estas interpretaciones, en muchos casos amañadas, perjudican y generan información incorrecta en contra de otros esfuerzos que buscan incrementar continuamente la calidad y la seguridad de los productos software.

La industria de TI es relativamente joven, de hecho sólo ha sobrevivido a un par de generaciones; pero también es una especie de mina de oro que, comparada con otras industrias y de acuerdo con creencias muy generalizadas, es relativamente bien pagada, no requiere esfuerzos físicos excesivos y se ejerce sin requisitos mínimos de entrada. Esto significa que economías enteras han crecido viendo a las TI como un juego de números: *si acumulas suficientes personas alrededor de un problema es posible que desaparezca, y si esa cantidad no genera costos elevados entonces se puede acumular un montón mayor*. Por esto es que, desde una perspectiva puramente demográfica, es posible afirmar que la mayoría de personas está en esta industria porque: 1) se trata de un trabajo bien remunerado, en comparación con otros empleos de carácter intelectual o incluso manual, o 2) no hay incentivos para hacer algo diferente a jugar cómodamente a los números.

Claro que también existen aquellos que optan por sobresalir construyendo buenos productos software. Ellos entienden que desarrollar bien es una habilidad, de hecho, toda una gama de habilidades: comprender y modelar el dominio del problema, entender y aplicar los lenguajes de programación, las librerías, los paradigmas, los idiomas, elegir qué aplicar en una situación dada, aprender y desarrollar algoritmos, comprender y dominar las fases de la Ingeniería de Software, automatizar procesos, conocer las teorías esbeltas para el suministro, la producción y el desarrollo de productos, aplicar la concurrencia y el paralelismo, destacar las bondades y potencialidades de las metodologías recientes, y así podría seguir la lista. Estos profesionales se quieren diferenciar de alguna manera, pero existen algunos interrogantes: ¿cómo potencializar sus habilidades y capacidades? Y ¿cómo pueden ayudar a otros en la industria a que tengan verdadero aprecio por el software que escriben? La respuesta puede ser que se necesita algún tipo de modelo de formación y una manera de identificar estas fortalezas, tanto entre los principiantes como en los experimentados. La cuestión es que en muchos casos el software se valora por la utilidad que proporciona, y no importa lo *feo* que sea internamente, siempre y cuando se entregue a tiempo. Un desarrollador de software puede construir un producto que *enamora* desde sus interfaces de usuario, pero lo que verdaderamente importa es que su interior, el código, se haya construido de acuerdo con las necesidades del cliente y con la seguridad y fiabilidad necesarias. *Ese es el producto de un desarrollador profesional*.

### 3.1 Desarrollar Software es una Ciencia y un Arte

Desarrollar software es tanto una ciencia como un arte. Es ciencia, porque en los procesos se deben considerar los principios de las Ciencias Computacionales y de la Ingeniería, y es arte, porque son tantas y tan diversas las variables involucradas en el desarrollo de productos software que no se puede tener solamente un proceso totalmente prescriptivo y repetitivo. De acuerdo con Brooks [1], en el desarrollo de software se articulan aspectos *esenciales* que se corresponden con los elementos inherentes y abstractos relacionados a la naturaleza del software, y aspectos *accidentales* correspondientes a las dificultades para llevar a cabo el desarrollo mismo, debido a las complejas relaciones humanas de comunicación y colaboración, es decir, a lo que Sommerville [2] clasifica como sistemas socio-técnicos.

En el arte, los productos tienen una belleza intrínseca en sí mismos. Una catedral realmente es una cabaña grande para que las personas se reúnan a orar; generalmente se construye de piedra para que dure más que una cabaña de madera, pero ¿por qué todo el material decorativo es de lujo? Por supuesto que está ahí para generar un sentido de grandeza y de imponencia, y llama la atención de aquellos que aprecian la belleza y magnificencia, por lo que entran con reverencia y humildad, listos para el culto. Lo que la hace arte es el trabajo y la belleza estética, por encima de su utilidad intrínseca y de la belleza estética. Entendemos el arte como un proceso creativo individual o colectivo, una profesión disciplinada donde los autores impregnan en la creación más que su intelecto, suman su alma y su impronta.

Existe una diferencia entre la mentalidad de un escultor, que esculpe la expresión en la cara de una gárgola, y un maestro de obra que utiliza bloques básicos para construir un parqueadero de varios pisos. En este caso, lo que menos se necesita es que el maestro esculpa su personalidad en la obra, de tal forma que los bloques tengan diferentes tamaños y que no sean intercambiables, lo que interesa es la funcionalidad. Cuando esto sucede, es decir, cuando el maestro de obra pierde de vista el objetivo de lo que está construyendo y coloca su firma, puede que logre una magnifica representación artística, pero que no responde a las necesidades del cliente. Entendemos que la Ingeniería de Software necesita más que esto, porque por sobre todas las cosas el arte, como cierto y de calidad, implica un compromiso y un resultado que trascienda aspectos básicos y cotidianos. Esto no implica dejar de lado las cuestiones técnicas ni la predominancia de la funcionalidad, pero el arte es una manera de hacer las cosas a nuestro entender y no sólo la búsqueda de la belleza, que es un concepto temporal y altamente subjetivo.

En los años 60, la comunidad del software aplicaba metáforas mecánicas para el proceso de desarrollo de software, pero actualmente la Ingeniería de Software es un área académica aceptada y un campo activo de investigación para las universidades y la industria. La perspectiva para el desarrollo de proyectos de esta ingeniería se define hoy como la aplicación de un enfoque disciplinado, cuantificable y sistemático, para desarrollar, operar y mantener productos software, es decir, es la aplicación de la ingeniería al software [3]. Este enfoque ha demostrado ser eficaz en el desarrollo de sistemas críticos de seguridad.

El software suficientemente bueno es una extensión lógica de las ideas de la Ingeniería de Software y de quienes la practican, pero que tienen un componente artístico. Es una labor de ingeniería en la que se intercambian recursos, calendarios, características y defectos. Por ejemplo, en el software del transbordador espacial la seguridad es una cuestión crítica, por lo que se tienen que minimizar los defectos, y las cuestiones artísticas pasan a segundo plano; pero las aplicaciones comerciales, como navegadores y editores gráficos, requieren una cantidad de características que se deben desarrollar rápidamente, la seguridad no es tan incidente pero asumen protagonismo cuestiones como combinación de colores, tipos de letra y la distribución del escritorio, es decir, la parte artística, además, los recursos son limitados por la necesidad de obtener un beneficio, por lo que su desarrollo se realiza pensando en la reducción de costos y de tiempo. Ambos enfoques de trabajo generan un mismo producto, software, pero con niveles diferenciados de seguridad, fiabilidad, consistencia, ciencia y arte. Por eso es que el desarrollo de software es tanto arte como ciencia, *es una profesión* con igual o mayores responsabilidades que en la Ingeniería Civil.

## 4. PROFESIONES Y PROFESIONALES

En este proceso lo primero es comprender qué es una profesión y qué es un profesional, aunque la búsqueda de una definición universalmente aceptada para la primera es en sí misma una empresa prácticamente imposible. Mientras que las sociedades occidentales apuntan a la medicina como el ejemplo más sólido de profesión, otras no tienen una visión unificada. El propósito no es entrar en discusiones sin sentido, sino aclarar algunos conceptos de acuerdo con la opinión generalizada.

El diccionario define *profesión* como una ocupación, especialmente una que requiere formación avanzada. Desafortunadamente, utilizar el término como sinónimo de *ocupación* es superficial y simplista, y no aporta beneficios para el objetivo de este documento. Otros la definen como una *vocación* cuya práctica se basa en la comprensión de la estructura teórica de algún área del saber o de la ciencia, y sobre las habilidades que la acompañan, mientras que otra definición dice que son grupos que aplican *conocimientos especiales* al servicio de un cliente. Estas últimas definiciones presentan deficiencias, porque casi todas las ocupaciones modernas basadas en servicios involucran algún conocimiento teórico, y al calificarlas como profesiones pierden su significado distintivo. Una profesión tiene que ser algo más que una reunión de proveedores de servicios.

En los años 70 se intentó definir el término con base en el control que tenían las asociaciones sobre sus miembros y el mercado, y creían que la autonomía profesional era un índice útil para valorar la situación relativa de las diversas ocupaciones profesionales. El consenso era que la distinción más estratégica reside en la autonomía legítima y organizada, y que una profesión era diferente de otras en que se le ha dado el derecho de controlar su propio trabajo. Posteriormente, esta cuestión tomó un enfoque diferente, cuando se discutió si la odontología podía ser considerada como una profesión, y entonces se argumentó que una característica común a todas las profesiones es la *promesa de altruismo que hacen con la sociedad*. La asociación de una profesión con la autoridad moral y el bien público se basa en que los profesionales tienen el deber, no sólo el derecho, de hablar sobre cuestiones relacionadas con su experiencia para alentar a los ciudadanos a tomar decisiones informadas.

Otra definición, que tiene amplia acogida, dice que una profesión es un colectivo de proveedores de servicios

expertos, que conjunta y públicamente se comprometen a priorizar las necesidades existenciales y los intereses del público, al que sirven sobre los suyos propios, y que a su vez reciben la confianza de éste para que lo hagan. Cuando una ocupación es reconocida como una profesión se firma un contrato social entre sus miembros y el público en general, y se sustenta en la confianza, el respeto, la condición social, el monopolio de la práctica y el derecho de auto-gobierno. Otra manera generalizada de definir una profesión es mediante el estudio de las características de las ya reconocidas, y examinar si la ocupación que se evalúa posee los mismos rasgos. Este enfoque, también conocido como de los rasgos o inventarios, se ha ampliado y utilizado para clasificar las ocupaciones en profesiones o semi-profesiones mediante el análisis a la ausencia o presencia de ciertos atributos. Desafortunadamente, este método tiene una desventaja evidente: los atributos se pueden convertir en una cortina de humo que sólo permite observar la apariencia y no el fondo de la cuestión.

Se observa entonces que el término *profesión* tiene varias definiciones: 1) es una *vocación* que requiere conocimiento especializado y una larga e intensiva preparación académica, y su práctica se basa en una comprensión de la estructura teórica de alguna área científica y de las habilidades que acompañan esa comprensión, 2) es una *capacidad* principal, una vocación, o un empleo y 3) es todo el *cuerpo de personas* que participan en una vocación. Por desgracia, la segunda definición describe con mayor precisión lo que actualmente se acepta para la Ingeniería de Software. Sin embargo, un creciente número de organizaciones, personas e instituciones están convencidas de que la primera definición es la más apropiada, y están trabajando por lograr su aceptación como tal. Estos movimientos han descubierto que un enfoque ingenieril para el ciclo de vida del software es menos caótico, produce mejores productos y también es rentable.

Ahora bien, ¿qué pasa con el término *profesionales*? En pocas palabras, los profesionales son miembros de una profesión reconocida, independientemente de si individual o colectivamente se comportan profesionalmente. Pero una mirada más profunda a esta definición, aparentemente obvia, revela dificultades elementales: la medicina es una profesión, pero ¿todos los practicantes de la medicina son automáticamente profesionales? Los médicos que tienen sus licencias suspendidas, pero que continúan en ejercicio y diagnostican pacientes y prescriben medicamentos dejan serias dudas acerca de si encajan como profesionales. Hay quienes piensan que debe ser la sociedad quien decida qué ocupaciones califican como profesiones, pero en lo que tiene que ver con la responsabilidad social y el bienestar de las comunidades debe ser el Estado el que decida acerca de estas cuestiones. Esto se puede lograr a través de licencias o certificaciones que emita un organismo oficial o un consejo independiente, las cuales, dentro de una profesión reconocida, trazan la línea entre los profesionales de confianza y los practicantes.

Entonces, al igual que el término *profesión*, un profesional puede ser: 1) alguien que se dedica a una ocupación o actividad profesional, es decir, alguien que se ajusta a los estándares técnicos y éticos de una profesión y 2) persona que ejerce una actividad por la cual recibe un beneficio económico. Si se tiene alguna duda acerca de que la segunda definición es la que refleja con mayor precisión a la mayoría de profesionales del software de hoy, basta con preguntarle a alguno de ellos acerca de las normas técnicas o éticas de su *profesión*.

### 4.1 Profesionalizar

Típicamente se acepta que profesionalización es la organización de una ocupación dentro de la forma asumida por los paradigmas de profesiones establecidas, como la medicina. En otros términos, es el proceso para promover una ocupación a una profesión, en el que se busca mejorar las habilidades de una persona con el fin de hacerla más competitiva en asuntos relacionados con su área de conocimiento. Los beneficios se reflejan al trasladar una serie de recursos —especialmente conocimientos y habilidades— a otros, que originarán recompensas sociales y económicas —para los profesionales—. A medida que una profesión adquiere mayor movilidad social y control de mercado podrá reflejar sus habilidades, experiencias y normas éticas. Esto se logra con productos fiables y seguros, que les permiten a sus miembros lograr el reconocimiento de la sociedad y afianzar su labor como profesión reconocida.

El trabajo de los profesionales se basa en proyectos, lo que implica que deben tener un principio y un fin claros, y un proceso metodológico de acciones deliberadas, planeadas e implementadas por los integrantes de los equipos. Esto es precisamente lo que se propone desde este Manifiesto para el desarrollo de software, porque el fin es obtener las recompensas que acompañan a la condición de ser profesión. Para lograrlo, el primer paso consiste en llegar a un consenso acerca de su cuerpo de conocimiento, trabajo que se viene desarrollando desde varias organizaciones. La legitimación de la autoridad profesional consta de tres peticiones distintivas: 1) que el conocimiento y la competencia de los profesionales hayan sido validados por una comunidad o por sus pares o iguales, 2) que este conocimiento validado descanse en fundamentos racionales y científicos y 3) que el juicio profesional y el asesoramiento estén orientados hacia un conjunto de valores sustantivos, como la salud en el caso de la medicina. Estos aspectos de legitimidad corresponden a los atributos colegiales, cognitivos y morales, incorporados en el término *profesión*.

El proceso de profesionalización se realiza en una serie de fases que no siempre se ejecutan de forma secuencial, es decir, algunas pueden ocurrir simultáneamente y otras lo hacen fuera de orden, pero siempre aprovechando el trabajo previo. La siguiente es una lista de esas fases:

1. *Reconocer la necesidad de la profesionalización*. Esto suele ocurrir cuando se tiene y se siente la necesidad de los individuos altamente calificados y uniformemente capacitados, es decir, cuando el

trabajo que realizan quienes ya están en esa área del conocimiento se convierte en fundamental y crítico en la naturaleza de las sociedades.

2. *Definir formalmente la profesión*. Esto incluye establecer una terminología normalizada para la profesión, creación de títulos y descripciones para los miembros, y la identificación de relaciones, por ejemplo, profesión con profesión, profesional con cliente y profesión con público en general. Algo en lo que se pueden encajar iniciativas como SWEBOK.

3. *Identificar asociaciones profesionales clave*. Se espera que estas sociedades hayan alcanzado cierto grado de estatus formal, por ejemplo, el respeto de la comunidad profesional, la publicación de buenas revistas y la realización de reuniones locales y nacionales. Además, serán muy valiosas para avanzar en el resto del proceso de la profesionalización.

4. *Establecer criterios mínimos de ingreso*. Esto debe incluir temas como cierto grado de formación formal, experiencia demostrable y una certificación. La mayoría de asociaciones en el mundo tienen sus reglamentos bien establecidos.

5. *Establecer y reconocer programas de formación y de entrenamiento*. Que incluyen programas de estudios universitarios existentes y programas profesionalizantes de formación continua aprobados. Los Ministerios de Educación y los organismos de control tienen esto cubierto.

6. *Establecer procedimientos formales de certificación y renovación*. El proceso de certificación se debe basar en la mínima cantidad de habilidades y conocimientos útiles que necesita un profesional tipo. El proceso de renovación de la certificación se debe dirigir hacia las trayectorias profesionales de los asociados. Obviamente, los procesos de certificación y formación se afectan directamente el uno al otro. Para la Ingeniería de Software, tal vez sea la fase que requiere mayor trabajo en este momento.

7. *Recolectar y promulgar estándares profesionales*. Esto estándares cubren temas como procedimientos, metodologías, métricas, niveles aceptables de rendimiento y herramientas. Actualmente, se tienen diversas propuestas y se requiere unificarlas y aceptarlas ampliamente.

8. *Identificar y divulgar un código de ética profesional*. Se establece un código de ética que deben respetar todos los miembros. Lo ideal sería que ese código se incorporará en los procesos de certificación. Para la Ingeniería de Software ya existe un código considerablemente aceptado.

9. *Establecer objetivos de rendimiento cuantificables para la profesión y los profesionales*. Se espera que los profesionales satisfagan y superen un conjunto de objetivos de rendimiento para mantener su estatus profesional, para lo que es necesario conocer lo bien que se están desempeñando en su área de conocimiento. La profesión se debe esforzar continuamente para mejorarse a sí misma como un todo, por ejemplo, disminuyendo la tasa de error promedio por miembro. Ya existen criterios como las buenas prácticas que se podrían estructurar para alcanzar este objetivo.

10. *Identificar los requisitos y procesos de formación continua*. Las profesiones no son estáticas, especialmente las relacionadas con TI, y la vida útil de los conocimientos profesionales continúa disminuyendo. Por ejemplo, se dice que el conocimiento técnico de la raza humana se duplica cada seis meses, por lo que un profesional está obligado a tomar continuamente cursos de actualización para mantener su estatus. El protagonismo aquí lo deben asumir las universidades, porque la Ingeniería de Software requiere especializaciones y cursos de actualización permanente.

11. *Identificar las responsabilidades y obligaciones profesionales*. Los profesionales deben ser conscientes de sus responsabilidades para con sus clientes, sus empleadores, otros profesionales y con la profesión en general. Además, deben ser consecuentes con las obligaciones que esas responsabilidades generan. Sólo las personas que están legalmente dementes no son responsables por sus acciones. Será el Estado quien legisle a este respecto.

12. *Establecer mecanismos legales de ingreso y de permanencia*. El estado de una profesión se ve disminuido por la inclusión de personas que no cumplen con las normas establecidas y por las violaciones que sus asociados cometen. El Estado debe reglamentar este ejercicio profesional, que a su vez influye para que las asociaciones delimiten los mecanismos.

13. *Establecer y mantener una imagen pública positiva*. Una profesión con una imagen pública positiva pueden conseguir mejores beneficios para sus miembros, incluidos niveles salariales más altos. Se debe hacer público el trabajo y los logros de los profesionales en Ingeniería de Software, tanto positivos como los errores, para que la sociedad mantenga una imagen reciente de los beneficios de esta profesión.

Obviamente, aunque lograr todo esto toma tiempo, se puede llevar a cabo en un período relativamente corto, por ejemplo, unos tres o cuatro años, tiempo que se puede reducir mediante un esfuerzo concentrado de parte de los propios profesionales. En el caso de la Ingeniería de Software, y como se evidencia en las fases, ya se han hecho importantes avances en pro de la profesionalización.

### 4.2 Recomendaciones para la profesionalización

La Ingeniería de Software se ocupa de la creación y de la aplicación de fundamentos ingenieriles para el desarrollo de software y del uso intensivo de productos técnicos, procesos que se logran mediante un análisis sistemático y trabajo en equipo. El concepto de la formación de ingenieros de software profesionales requiere sujetos, secuencias sistemáticas y flexibilidad para hacerle frente a las necesidades del mercado, los requisitos de la industria y los rápidos y abundantes desarrollos tecnológicos. Uno de los objetivos de este Manifiesto es integrar conceptos como formación, formalización y profesionalización en todos los ámbitos relacionados con el desarrollo de software. Aunque no se puede aseverar que el simple seguimiento de un algoritmo sea suficiente para alcanzar la profesionalización de un área de conocimiento, a continuación se plantean unos pasos que pueden ayudar a lograrlo:

1. *Iniciar el proceso de una vez*, con el objetivo de alcanzar resultados positivos visibles en no más de dos años. Específicamente, las funciones del desarrollador y del administrador promedio se deben ver afectadas por el proceso de profesionalización antes de esa línea de tiempo.

2. *Invitar a muchos y diversos actores a participar y a opinar*. Incluyendo a desarrolladores, analistas, gestores, profesores, gobierno, sociedades profesionales y a profesiones establecidas, entre otros.

3. *Dar a conocer el progreso del proceso*, y mantenerlo visible y asequible para todos los interesados, de forma que con sus opiniones y aportes se enriquezca y perfeccione en todo momento.

4. *Establecer un comité de mantenimiento para la profesionalización*. El trabajo de este comité incluirá el seguimiento a los cambios en el proceso, introducir estos cambios de manera ordenada y actuar como un *tribunal supremo* de las controversias que surjan.

5. *Fomentar el establecimiento de planes de estudio* para el programa de Ingeniería de Software. Por otra parte, promover el concepto de profesionalización en todos los niveles de formación relacionados con el software.

## 5. PROFESIONALIZAR EL DESARROLLO DE SOFTWARE

El software es el elemento clave y se ha involucrado en los principales desarrollos tecnológicos. La rápida evolución de la tecnología informática ha propiciado la definición de nuevos servicios y esquemas de trabajo en las organizaciones, y les ha permitido mejorar la calidad de servicio a sus clientes, definir nuevos servicios, conquistar nuevos mercados, y en general ser más competitivas. La operación de las compañías se encuentra cada vez más soportada en sistemas de información intensivos en software, que son fundamentales para apoyar su liderazgo estratégico en el mercado o, por el contrario, propiciar su fracaso [4]. Igualmente, el software ha incursionado en los campos del entretenimiento y el hogar, modificando el estilo de vida y abriendo posibilidades a nuevas formas de trabajo y nuevos modelos de negocio.

La Ingeniería de Software es una disciplina relativamente joven con respecto a otras ingenierías, pero al igual que para las demás, su reto es solucionar problemas en un mundo en constante cambio, y lograr una adecuada relación costo-beneficio, aplicando principios matemáticos y de las Ciencias Computacionales [5]. Las características particulares del producto software, la complejidad y la dinámica del contexto, y la rápida evolución de la electrónica y estas Ciencias, le imponen una complicación particular a la disciplina, lo que la diferencian de las demás [6, 7). Hoy en día, el software es reconocido como un sistema socio-técnico que comprende uno o más sistemas, pero crucialmente, también incluye conocimiento acerca de cómo utilizarlo para alcanzar un objetivo más amplio. Este tipo de sistemas incluye a las personas como partes inherentes del mismo, son gobernados por políticas y reglas organizacionales, y se pueden ver afectados por restricciones externas, como las leyes nacionales y las políticas reguladoras [8].

Al analizar el estado actual de la Ingeniería de Software se podría tomar dos posturas: 1) la *pesimista*, planteada desde la conferencia de la NATO en 1968, cuando se habló por primera vez de la *crisis del software*, y que, aún hoy, muchos consideran que no se ha superado debido los resultados poco halagadores que con frecuencia se observan en los proyectos: incumplimiento sistemático de los plazos de entrega, desfase en los tiempos y presupuestos, y baja calidad del producto final. Pressman [8] define esta problemática como *aflicción crónica*, pero afortunadamente existen movimientos que buscan solucionarla, como el SEMAT. 2) La *optimista,* que sin pretender desconocer las problemáticas que actualmente se presentan, busca enfrentar la alta complejidad del desarrollo de software en entornos altamente cambiantes, entiende los desafíos propios de los sistemas socio-técnicos, y reconoce la evolución positiva y los resultados logrados por la Ingeniería de Software hacia su madurez. Tal como lo planteó Ambler [9] en Zurich: "*Somos más éxitos de lo que pensamos; la definición de éxito de un proyecto debe evaluarse en función de su contexto; cada organización define sus criterios de éxito del proyecto que van más allá de los criterios de tiempo, presupuesto y alcance*".

Actualmente no es posible concebir el mundo moderno sin software. Las infraestructuras nacionales y todas sus utilidades están controladas por sistemas basados en computadores, y los productos eléctricos de control incluyen un procesador y el software necesario. La fabricación y distribución industrial está completamente informatizada, y la industria del entretenimiento, como la música, los video-juegos, el cine y la televisión, utilizan software intensivo. Por lo tanto, la Ingeniería de Software es esencial para el funcionamiento de la sociedad de este siglo. Pero, debido a que los sistemas software son abstractos e intangibles, a que no están limitados por las

propiedades de los materiales, ni gobernados por las leyes físicas o por los procesos de fabricación, se pueden convertir en extremadamente complejos, difíciles de entender, y costosos de modificar.

Aunque muchas personas en diferentes contextos escriben programas, como en los negocios para simplificar el trabajo, los científicos e ingenieros para procesar sus datos experimentales, y los aficionados para su propio interés y disfrute, esto no las califica como desarrolladores, o ni siquiera programadores en muchos casos. Además, la mayor parte del desarrollo de software es una actividad profesional en la que los productos se desarrollan para fines comerciales específicos, para incluirlos en otros dispositivos, o como productos de los sistemas de información, que están destinados a ser utilizados por alguien diferente a su creador, y que generalmente los desarrollan equipos y no individuos.

Esta situación ha hecho que muchos piensen que *software* no es más que otra palabra que se relaciona con los computadores. Sin embargo, cuando se habla de Ingeniería de Software, el término no sólo se relaciona con programas, sino también con la documentación asociada y con los datos de configuración que se requiere para hacer que los programas funcionen correctamente. A menudo, un sistema software es la combinación de varios programas, porque suele consistir de una serie de componentes independientes y de archivos de configuración, utilizados para implantarlos en un entorno de trabajo. Es por esto, que si alguien escribe un programa por sí mismo, que nadie más va a usar, no tendrá por qué preocuparse de escribir las guías de uso, ni la documentación de diseño.

Sin embargo, si se está escribiendo software que otras personas utilizarán y otros ingenieros modificarán, por lo general se tendrá que proporcionar información adicional, además del código del programa. Este es un principio de las actividades de Verificación y Validación, porque al hablar de la calidad de software, se deba tener en cuenta que el producto lo utilizan y modifican personas diferentes a sus desarrolladores. La calidad no sólo tiene que ver con lo que hace el software, sino que también incluye el comportamiento mientras se ejecuta, y la estructura y organización de los demás programas del sistema, lo mismo que la documentación asociada.

Generalmente, los desarrolladores de software adoptan un enfoque sistemático y organizado para realizar su trabajo, porque es la manera más eficaz para producir software de alta calidad. Sin embargo, los programadores parecen no acercarse a seleccionar un método adecuado, y a veces ni siquiera lo tienen en cuenta, lo que genera una serie de circunstancias a través de *interpretaciones* poco creativas, menos formales y más rápidas, de las propuestas existentes, que aplican en sus proyectos. El desarrollo profesional de software es importante por dos razones:

1. Los individuos y la sociedad dependen cada vez más de sistemas software avanzados, por lo que se requiere suficiente capacidad para producirlos de forma fiable, segura, eficiente y fidedigna.

2. Generalmente, a largo plazo es más económico aplicar métodos y técnicas de Ingeniería de Software para desarrollar sistemas en lugar de sólo escribir programas, como si fuera un proyecto personal. Para la mayoría de sistemas, los costos más elevados tienen que ver con las actualizaciones y modificaciones posteriores a la implementación.

Por todo lo expuesto hasta el momento, es que mantenemos nuestra posición de que se necesita profesionalizar el desarrollo de software, y porque, a diferencia de otras actividades, parece no tener una regulación adecuada en cuanto a requisitos de empleabilidad de sus practicantes, ni de las responsabilidades sociales, y mucho menos de una formación acorde con las exigencias de este siglo. El propósito de este reporte es describir la situación, los objetivos y el alcance de la iniciativa de profesionalización del desarrollo de software. Este informe actúa entonces como un manifiesto para la colectividad, y aclara el papel que queremos desempeñar, tanto en la comunidad científica, como en la académica, la industrial y el Estado. Aquí es importante aclarar nuestra comprensión acerca de la Ingeniería de Software, ya que es un término con amplio uso pero que tiene diferentes significados para cada persona y situación. En pocas ocasiones la visión global de la Ingeniería de Software es completamente positiva, por lo que es un área temática y una profesión que sufre de una sobrecarga de tecnología y de falta de rigurosidad histórica.

Para nosotros, *la Ingeniería de Software tiene por objeto mejorar la práctica del desarrollo de software de alta calidad*. Esta área del conocimiento es el centro de las Ciencias Computacionales, porque generar software de alta calidad para correr de forma eficiente en el hardware es el objetivo principal de la investigación en este campo. Paradójicamente, y debido a su importancia, puede ser difícil identificar lo que es exactamente, y sobre todo, la forma en que se diferencia de otras áreas de investigación de estas Ciencias. Sin embargo, creemos firmemente que todas las áreas de las Ciencias, especialmente aquellas en las que el manejo de información es esencial, están altamente relacionadas con la Ingeniería del Software y sus productos. Además, la sociedad tiene una interacción directa con todo tipo de sistemas y software —de ahí la denominación de *software-dependiente*—, por lo que la calidad y la fiabilidad de estos productos son la piedra angular de un buen desarrollo de software. Pero la Ingeniería de software se ha convertido en el *patito feo* de nuestro oficio, ya que es invitada a las reuniones familiares pero nunca a una noche de paseo por la ciudad. Creemos que varias causas han originado y enfatizado esta situación, como el hecho de que gran parte de la industria y de los programadores de software han interpretado a su amaño el *Manifiesto para el Desarrollo Software Ágil*, y se olvidan de las ventajas que esta iniciativa le aporta al logro de productos a tiempo y en el presupuesto. Algunas de estas interpretaciones son:

1. Lo mejor es saltar inmediatamente a escribir código y dejar de preocuparse por escribir requisitos y especificaciones de diseño completos. Esta es una *buena práctica* teniendo en cuenta que la mayor parte del código que escriben, al faltarles formación adecuada, no tiene el arte ni la esencia científica que requiere para la calidad y la fiabilidad. De todos modos, siempre quisieron hacer las cosas a su manera, e interpretan que existe un principio que les dice que está bien.

2. No tenemos que perder el tiempo escribiendo documentos para explicar lo que estamos haciendo. En todo caso, sino comprenden lo que necesitan, entonces sobra el material técnico.

3. Estamos seguros de que nuestros empleadores estarán encantados de recibir lo que producimos, porque les entregamos a tiempo, cuando antes no sabían si iban a conseguir nada en absoluto.

4. Siempre admiramos a aquellos raros *desarrolladores* ingeniosos que pueden *hackear* nuestras bases de código y escribir otro tan brillante que nadie siquiera lo puede comprender. Ahora tenemos un nuevo nombre para ellos: *codificadores ágiles* (como parodia).

En caso de que no haya quedado claro hasta el momento, estas interpretaciones son muy, muy perjudiciales para la calidad de los productos software. Las personas creen que la Ingeniería de Software trata sólo y exclusivamente de dominar la sintaxis y las *APIs* de un lenguaje, y que la calidad y la tasa de éxito de nuestros proyectos sigue siendo el mismo año tras año. Nosotros creemos que se debe considerar cada metodología y cada principio que aporte al logro de productos de calidad, pero dentro de los lineamientos de un profesión en proceso de maduración, y no como espejos para distraer del objetivo de desarrollar software serio, responsable y que cumpla los requisitos del cliente y de la sociedad.

Por todo esto estamos de acuerdo con lo que expresa McConnell [10], acerca de que las prácticas necesarias para crear buen software fueron establecidas y están fácilmente disponibles desde hace más de 20 años, pero han sido manipuladas e interpretadas a la luz de intereses personales, y los productos resultantes, a pesar de algunos éxitos sorprendentes, no están a la altura de las necesidades. Existe un amplio abismo entre las prácticas normales y la mejor, y muchas de uso generalizado son peligrosamente obsoletas y de poca utilidad, por lo que el desempeño promedio de los proyectos software, que se trabajan bajo está lupa, deja mucho que desear y diversos desastres conocidos lo constatan. Por esto es que estamos trabajando para que la comunidad del software comprenda, aplique y experimente los principios de cada propuesta antes de decidir aplicarla sin razón, porque aunque las buenas prácticas están claras en la teoría desde hace años, pero la puesta en práctica, la contextualización y el compromiso con el que se llevan adelante son el problema principal, y es lo que en parte genera las crisis y las falencias de la industria Software, no las tendencias metodológicas en sí.

Por último, creemos que la Ingeniería de Software es una disciplina práctica, por lo que su investigación nos obliga a participar en el trabajo experimental. Por lo tanto, trabajamos con empresas y organizaciones en proyectos reales, a menudo como parte de proyectos de investigación financiados por la academia, la industria o el Estado, pero también en proyectos de consultoría y de formación, incluso dentro de las mismas empresas. Esto nos permite estar en sintonía con sus problemas y poner a prueba nuestras ideas en el campo, porque sólo la investigación que se valida en el mundo real se puede decir que demuestra su eficacia.

## 6. CONCLUSIÓN

**MANIFIESTO POR LA PROFESIONALIZACIÓN DEL DESARROLLO DE SOFTWARE**

**Debido a que**

EL SOFTWARE ES UN PRODUCTO DEL INTELECTO HUMANO, QUE PARA SU PRODUCCIÓN REQUIERE INGENIERÍA, CIENCIA, GESTIÓN, IMAGINACIÓN Y ARTE

LA COMPLEJIDAD DEL SOFTWARE, DEL PROCESO PARA SU DESARROLLO Y DE LAS NUEVAS Y VARIADAS NECESIDADES DE LA SOCIEDAD RESPECTO A SUS PRODUCTOS SE INCREMENTA CONSTANTEMENTE

EL SOFTWARE TIENE CADA DÍA MAYOR PROTAGONISMO EN LAS ACTIVIDADES COTIDIANAS DE LAS PERSONAS, Y GRAN PARTE DE ELLAS DEPENDE DE SU CORRECTO FUNCIONAMIENTO

EL DESARROLLO DE SOFTWARE ES UNO DE LOS SECTORES ESTRATÉGICOS DE LA ECONOMÍA QUE LOS ESTADOS PROMUEVEN Y APOYAN

LA INDUSTRIA NECESITA PROFESIONALES ALTAMENTE CAPACITADOS PARA RESPONDER A LAS NECESIDADES DE LA SOCIEDAD DE LA INFORMACIÓN Y EL CONOCIMIENTO

EL INGENIERO DE SOFTWARE DEBE CONTAR CON CONOCIMIENTOS Y COMPETENCIAS, GENÉRICOS Y ESPECÍFICOS, QUE LE PERMITAN DESARROLLAR SOLUCIONES DE CALIDAD E INTEGRADAS EN CONTEXTO

**Por lo tanto**

**SE NECESITA PROFESIONALIZAR EL DESARROLLO DE SOFTWARE**

*Estableciendo claramente las responsabilidades, capacidades, conocimientos, habilidades y formación de los ingenieros de software.*

## 7. REFERENCIAS


[1] Brooks, F. (1987). No Silver Bullet - Essence and Accident in Software Engineering. Computer, 20(4), pp. 10-19.

[2] Sommerville, I. (2010). Software Engineering. Addison-Wesley.

[3] IEEE. (1991). Standard Computer Dictionary. A Compilation of IEEE Standard Computer Glossaries. IEEE 610-1991.

[4] Anaya, R. (2006). Una visión de la enseñanza de la Ingeniería de Software como apoyo al mejoramiento de las empresas de software. Revista Universidad EAFIT, 42(141), pp. 60-76.

[5] Ford, G. (1990). SEI Report on Undergraduate Software Engineering Education. Technical Report: CMU/SEI-90-TR-003. Software Engineering Institute.

[6] Bruegge, B., Dutoit, A. (2009). Object-Oriented Software Engineering Using UML, Patterns, and Java. Prentice Hall.

[7] Maibaum, T. (1997). What We Teach Software Engineers in the University: Do We Take Engineering Seriously? Software Engineering Notes, 22(6), pp. 40-50.

[8] Pressman, R. (2009). Software Engineering: A Practitioner's Approach. McGraw-Hill Science.

[9] Ambler, S. (2010). Context Counts: Position Paper for SEMAT. Proceedings of the Semat, Zurich Workshop.

[10] McConnell, S. (2003). Professional Software Development: Shorter Schedules, Higher Quality Products, More Successful Projects, Enhanced Careers. Addison Wesley.