

# La Esencia de la Ingeniería de Software: El Núcleo de Semat

Ivar Jacobson, Pan-Wei Ng, Paul E. McMahon, Ian Spence y Svante Lidman  
Integrantes de la iniciativa Semat (Método y Teoría de la Ingeniería de Software)

Traducción de: Carlos Mario Zapata Jaramillo  
Facultad de Minas  
Universidad Nacional de Colombia  
Medellín, Colombia  
cmzapata@unal.edu.co

*Artículo publicado originalmente en Communications of the ACM, vol. 55, no. 12, december 2012, pp. 42-49. El traductor declara tener autorización de los autores para difundir la versión en español en la Revista Latinoamericana de Ingeniería de Software, sin que ello presente incompatibilidades o vulnere los derechos de Communications of the ACM.*

**Resumen**— Esta es una traducción del artículo que se publicó en *Communications of the ACM* en Diciembre de 2012 bajo el título “*The Essence of Software Engineering: The SEMAT Kernel*” y se realiza debido a la importancia que adquiere el tema para la audiencia de hispanoparlantes, de forma que se pueda generar un acceso a lo que constituye uno de los últimos adelantos en Ingeniería de Software.

**Palabras clave**— Núcleo, alfa, espacio de actividad, estado, Semat

## I. INTRODUCCIÓN

Todo aquel que desarrolla software sabe que es un negocio complejo y riesgoso y que sus participantes siempre buscan nuevas ideas que los conduzcan a desarrollar mejor software. Por fortuna, la ingeniería de software es aún una profesión joven y creciente que mira las innovaciones y mejoras en las mejores prácticas cada año. Por ejemplo, basta mirar las mejoras y beneficios que el pensamiento lean y ágil tuvo sobre los equipos de desarrollo de software.

Los equipos exitosos de desarrollo de software necesitan establecer un balance entre las entregas rápidas de sistemas de software que trabajen bien, la satisfacción de sus interesados, el tratamiento de sus riesgos y la mejora de sus formas de trabajo. Para eso, ellos necesitan un marco de pensamiento efectivo que cierre la brecha entre su actual forma de trabajo y cualquier idea nueva que quieran adoptar. Este artículo presenta ese marco de pensamiento en forma de un núcleo accionable, que podría beneficiar cualquier equipo que desee balancear sus riesgos y mejorar su forma de trabajo.

El trabajo en el núcleo, la esencia de la ingeniería de software, se inspiró y es una respuesta directa al llamado a la acción de los Métodos y Teoría de la Ingeniería de Software (Semat, por sus iniciales en inglés; nota del traductor: si bien el origen de la palabra Semat lo constituyen las iniciales de las palabras que conforman la sigla, acogiendo el espíritu de la iniciativa, se prefiere colocarlo con mayúscula inicial, pues se pretende que, más que un acrónimo, se convierta en un sustantivo), el cual es, a su modo, un pequeño paso hacia la redefinición de la ingeniería de software.

Ivar Jacobson, Bertrand Meyer y Richard Soley fundaron Semat en septiembre de 2009, cuando sintieron que había llegado el momento de cambiar fundamentalmente la forma en

que la gente trabaja con métodos de desarrollo de software [3, 4, 8]. Ellos escribieron una declaración del llamado a la acción, que en pocas líneas identifica una cantidad de problemas críticos, explica por qué se necesita actuar y sugiere qué se necesita para hacerlo. El llamado a la acción es:

Algunas áreas de la ingeniería de software hoy en día sufren prácticas inmaduras. Los problemas específicos incluyen:

- La prevalencia de bogas más típicas en la industria de la moda que en una disciplina ingenieril.
- La carencia de una base teórica sonora y ampliamente aceptada.
- La gran cantidad de métodos y variantes de métodos, con diferencias que poco se entienden y que se magnifican artificialmente.
- La carencia de evaluación y validación experimentales y creíbles.
- La separación entre la práctica industrial y la investigación académica.

La afirmación del llamado a la acción de Semat de que la industria de software es propensa a bogas y modas hace que la gente asuma que Semat se opone a las nuevas ideas. Esto no podría estar más lejos de la verdad. Como usted podrá ver en este artículo y en el libro que pronto se publicará, llamado “La esencia de la ingeniería de software: aplicando el núcleo de Semat” [6], los partidarios de Semat son muy entusiastas con las nuevas ideas. Ellos están en contra del comportamiento contrario a lo lean y a lo ágil. Este comportamiento proviene de gente que adopta soluciones inapropiadas porque creen que esas soluciones están de moda o porque los presionan sus pares o la corrección política.

Semat apoya un proceso para redefinir la ingeniería de software, basado en una teoría sólida, principios probados y mejores prácticas que:

- Incluyan un núcleo de elementos ampliamente aceptados y que se pueda extender a usos específicos.
- Traten asuntos tecnológicos y humanos.
- Los apoyen la industria, la academia, los investigadores y los usuarios.
- Apoyen la extensión ante los requisitos cambiantes y la tecnología.

El llamado a la acción de Semat recibió un amplio apoyo, incluyendo una lista creciente de signatarios y partidarios

(<http://www.semat.org>). En febrero de 2010, los fundadores de Semat desarrollaron el llamado a la acción por medio de una declaración de visión [5]. De acuerdo con esta visión, Semat se enfoca en dos objetivos principales: encontrar un núcleo de elementos ampliamente aceptados y la definición de una base teórica sólida.

En gran medida, esas dos tareas son independientes una de otra. Encontrar el núcleo y sus elementos es un ejercicio pragmático que requiere desarrolladores experimentados de software con conocimiento de muchos de los métodos existentes. Definir la base teórica es una investigación académica y podría tomar muchos años para conseguir un resultado exitoso.

## II. EL PODER DE UN TERRENO COMÚN

El primer paso de Semat era identificar un terreno común para la ingeniería de software. Este terreno común se manifestaba como un núcleo de elementos esenciales que son universales a todos los esfuerzos de desarrollo de software y un lenguaje sencillo para describir métodos y prácticas. El núcleo se publicó inicialmente en la entrega [2, 9] que hizo Semat al OMG (Grupo de Gestión de Objetos, por sus siglas en inglés). Como se muestra en las figuras 1 y 2, el núcleo contiene un pequeño número de “cosas con las que siempre trabajamos” y “cosas que siempre hacemos” cuando se desarrollan sistemas de software. También se está adelantando trabajo para definir las “habilidades que siempre necesitamos tener”, pero esto tendrá que esperar hasta futuras versiones del núcleo.

Más que un modelo conceptual, el núcleo provee:

- Un marco de pensamiento para que los equipos razonen sobre el progreso que están haciendo y la salud de sus esfuerzos.
- Un terreno común para la discusión, mejoramiento, comparación e intercambio de métodos y prácticas de ingeniería de software.
- Un marco para que los equipos ensamblen y mejoren continuamente su forma de trabajo, mediante la composición de prácticas definidas por separado y de diverso origen.
- Un fundamento para la definición de medidas que no dependan de las prácticas, para evaluar la calidad del software producido y los métodos que se usan para producirlo.
- Más importante aún, una forma de ayudarle a los equipos a comprender dónde están, qué deberían hacer luego y dónde necesitan mejorar.

## III. LA IDEA GENERAL

¿Qué hace del núcleo algo más que sólo un modelo conceptual de la ingeniería de software? ¿Qué es lo realmente nuevo acá? Esto se puede resumir en sus principios fundamentales (véase la Figura 3), que realmente suministran tres características únicas del núcleo: es accionable, es extensible y es práctico.

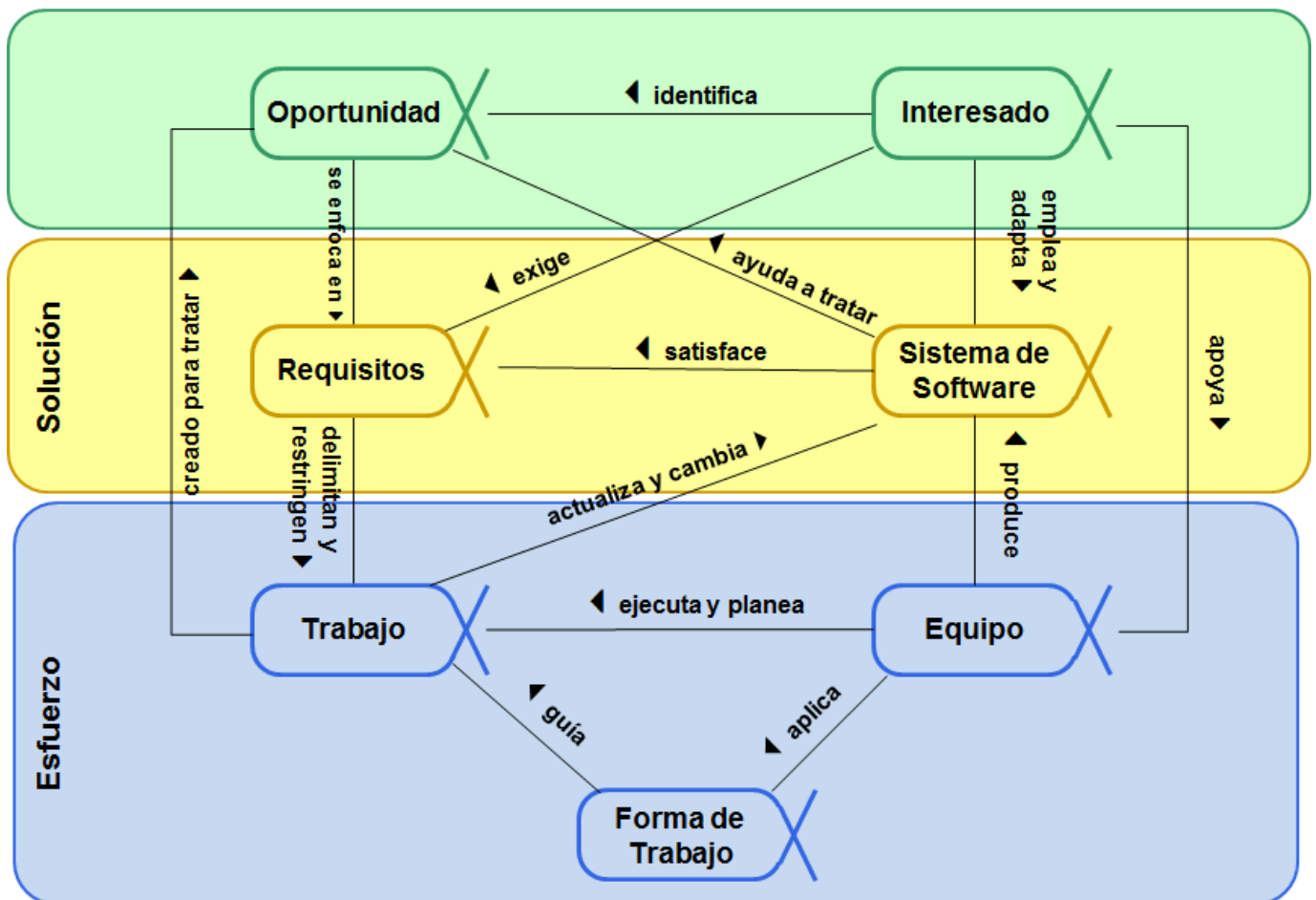


Fig. 1. Cosas con las que siempre trabajamos.

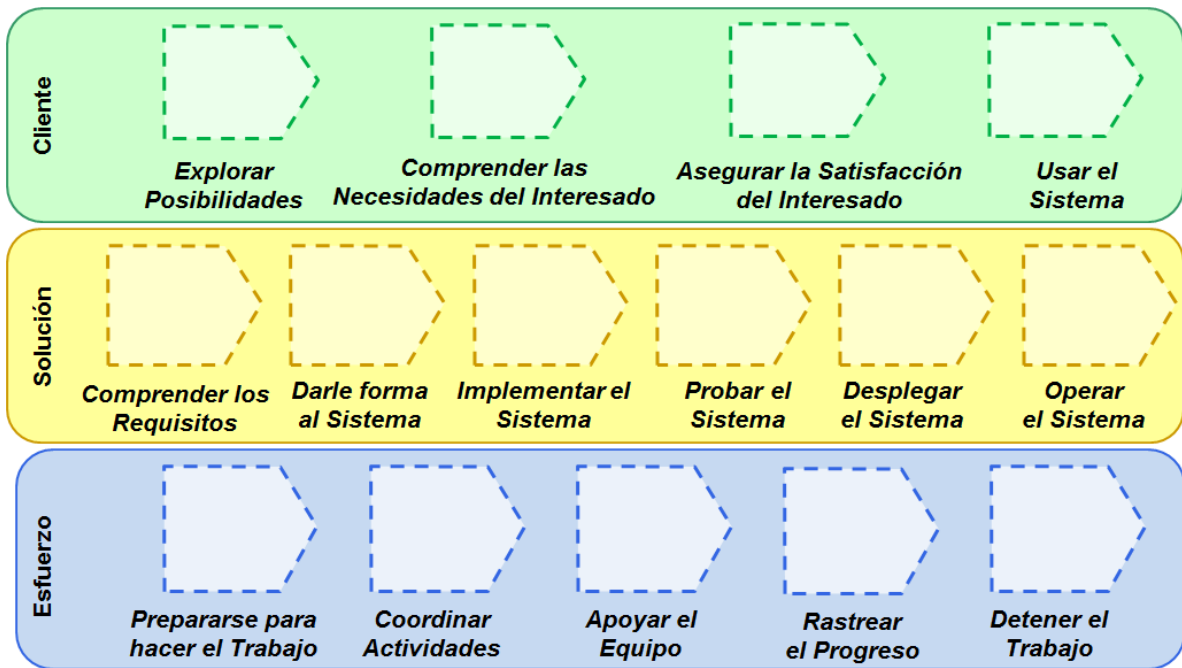


Fig. 2. Cosas que siempre hacemos.



Fig. 3. Principios fundamentales que apoyan el núcleo.

**El núcleo es accionable.** Una característica única del núcleo es la forma en que maneja las “cosas con qué trabajar”. Ellas se capturan como alfas en lugar de productos de trabajo (tales como documentos). Un alfa es un elemento esencial del esfuerzo de ingeniería de software, el cual es relevante para una evaluación de su progreso y salud. Como se muestra en la Figura 1, Semat identificó siete alfas: oportunidad, interesados, requisitos, sistema de software, trabajo, forma de trabajo y equipo.

Los alfas se caracterizan por un simple conjunto de estados que representan su progreso y salud. Por ejemplo, el sistema de software se mueve entre los estados “con arquitectura seleccionada”, “demostrable”, “usable”, “listo”, “operacional” y “retirado”. Cada estado tiene una lista de chequeo que especifica los criterios necesarios para alcanzar un estado. Esos estados hacen que el núcleo sea accionable y lo habilitan para guiar el comportamiento de los equipos de desarrollo de software.

El núcleo no presenta el desarrollo de software como un proceso lineal, sino como una red de elementos que colaboran y que se deben balancear y mantener, de modo que los equipos puedan hacer progresos efectivos y eficientes, eliminar desperdicios y desarrollar muy buen software. Los alfas en el núcleo proveen un marco general para dirigir y hacer progresar los esfuerzos de desarrollo de software, sin importar las prácticas que se apliquen o la filosofía que se siga.

A medida que se agregan prácticas al núcleo, se agregarán nuevos alfas para representar las cosas que dirijan o inhiban el progreso de los alfas del núcleo. Por ejemplo, el alfa “requisitos” no se tratará en conjunto, sino que avanzará ítem por ítem. El progreso de los ítems individuales de requisitos conducirá o inhibirá el progreso y salud del alfa “requisitos”. Los ítems de requisitos pueden ser de diferentes tipos, por ejemplo, características, historias de usuario o porciones de casos de uso. Cada tipo se puede representar como un alfa y tener unos estados a seguir. El beneficio de relacionar esos ítems pequeños con el núcleo de granularidad más gruesa radica en que permite seguir la salud del esfuerzo en conjunto. Esto proporciona un balance necesario al seguimiento de bajo nivel de los ítems individuales, permitiendo a los equipos entender y optimizar sus formas de trabajo.

**El núcleo es extensible.** Otra característica única del núcleo es la manera en que se puede extender para apoyar diferentes proyectos (por ejemplo, nuevos desarrollos, mejoras a sistemas legados, desarrollos internos, desarrollos externos, líneas de productos de software, etc.). El núcleo le ayuda a agregar prácticas, tales como historias de usuario, casos de uso, desarrollo basado en componentes, arquitectura, programación por pares, reuniones diarias de pie, equipos auto organizados y otras. Se buscan prácticas para construir los métodos que usted necesita. Por ejemplo, diferentes métodos se podrían ensamblar para desarrollo interno y subcontratado o para el desarrollo de sistemas embebidos de seguridad crítica y sistemas de reporte de asistencia administrativa.

La separación de prácticas es la idea clave aquí. Ya que el término “práctica” se usó ampliamente en la industria por muchos años, el núcleo tiene un enfoque específico en el manejo e intercambio de prácticas. Las prácticas se representan como unidades modulares separadas y distintas, que un equipo puede escoger usar o no. Esto contrasta con los enfoques tradicionales que tratan el desarrollo de software como una sopa de prácticas que no se diferencian y conduce a los equipos a deshacerse de lo bueno y lo malo cuando se mueven de un método a otro.

**El núcleo es práctico.** Quizá la característica más importante del núcleo es la manera en que se usa en la práctica. Los enfoques tradicionales de los métodos de desarrollo de software tienden a enfocarse en apoyar a los ingenieros de procesos o los ingenieros de calidad. El núcleo, en contraste, es un marco de pensamiento tangible y práctico que apoya a los profesionales de software a medida que realizan su trabajo.

Por ejemplo, el núcleo se puede “tocar” y usar [7, 10] empleando tarjetas (véase la figura 4). Las tarjetas proporcionan recordatorios concisos y señales para los miembros del equipo, a medida que realizan sus tareas diarias. Al proporcionar listas de chequeo y sugerencias prácticas, en lugar de discusiones conceptuales, el núcleo se convierte en algo que el equipo usa diariamente. Esta es una diferencia fundamental con los enfoques tradicionales, que tienden a exagerar el énfasis en las descripciones de los métodos, en lugar del uso de los métodos y, por ello, sólo los suele consultar gente nueva del equipo.

Las tarjetas proporcionan una descripción concisa que sirve como recordatorios para los miembros del equipo. Ellos pueden mantener el núcleo como un pequeño mazo de tarjetas en sus bolsillos, que fácilmente pueden extraer para discutir el estado actual del desarrollo, la asignación de trabajo y la colaboración entre los miembros del equipo. Los equipos también pueden discutir áreas de mejoramiento refiriéndose a las tarjetas. Así, el núcleo no es sólo una descripción pesada de lo que un equipo necesita hacer. En su lugar, forma una parte esencial de lo que hacen cada día.

**El núcleo en acción.** El núcleo tiene muchas aplicaciones en las vidas diarias de los profesionales de software. Algunas de ellas son:

- Correr una iteración o un *sprint*.

- Correr el desarrollo completo, desde la idea hasta el producto.
- Escalar hacia grandes organizaciones y esfuerzos complejos de desarrollo de software.

La primera aplicación, la planeación de una iteración, se usa acá como un ejemplo de lo que un equipo puede hacer con el núcleo. Las otras se cubren completamente en *La Esencia de la Ingeniería de Software: aplicando el núcleo de Semat*.

El ejemplo que se presenta acá asume que una compañía tiene muy poco en cuanto a procesos formales. En el pasado, se creía en tener individuos creativos y habilidosos en los equipos experimentados, pero la compañía no crecía ni tenía muchas nuevas contrataciones. Esos nuevos empleados, la mayoría recién salidos de la universidad, tienen buenas habilidades técnicas (por ejemplo en lenguajes de programación), pero no son tan habilidosos en otros aspectos del desarrollo de software, como el trabajo con los interesados para lograr acuerdos en los requisitos.

Esta compañía tiene un equipo de desarrollo que es responsable por hacer una aplicación móvil para redes sociales que permita a los usuarios compartir ideas, fotos y comentarios y navegar por ellos. El equipo comenzó con sólo dos desarrolladores, Smith (el líder del equipo) y Tom, los cuales tenían familiaridad con el núcleo. Luego, se les unieron otros dos desarrolladores, Dick y Harriet, que eran nuevos en el trabajo y no tenían experiencia previa en el núcleo. Para Smith, el líder del equipo, el éxito significaba más que funcionalidad, cronograma y calidad. Este equipo corría el desarrollo de forma iterativa. Usted puede pensar en planear una iteración de la manera siguiente:

1. Determine dónde está: resuelva el estado actual del esfuerzo
2. Determine a dónde ir: decida en qué enfocarse luego y cuáles serán los objetivos de la próxima iteración.
3. Determine cómo llegar allá: póngase de acuerdo en las tareas que el equipo necesita para lograr los objetivos.



Fig. 4. Las tarjetas hacen tangible el núcleo.



#### IV. DETERMINE DÓNDE ESTÁ EL EQUIPO UTILIZANDO EL NÚCLEO

Supongamos que Smith y su equipo llevan seis semanas en el desarrollo. Ellos proporcionaron una demostración temprana del sistema a sus interesados, quienes se sintieron satisfechos y proporcionaron una valiosa retroalimentación. Sin embargo, el sistema aún no lo pueden correr los usuarios finales. Usted puede usar el núcleo para hacer esto de varias formas. Si usted está empleando tarjetas de estado de los alfas, entonces usted puede hacer un tutorial como sigue:

1. Coloque las tarjetas de cada alfa en una fila de una tabla con el primer estado a la izquierda y el último estado a la derecha.
2. Deténgase en cada estado y pregunte a su equipo si se logró ese estado.
3. Si se logró el estado, mueva la tarjeta hacia la izquierda. Continúe con la siguiente tarjeta hasta que logre el estado que su equipo aún no logra.
4. Mueva esta tarjeta de estado y el resto de las tarjetas pendientes de estado hacia la derecha.

La Figura 5 muestra los estados que el equipo de Smith logró a la izquierda y aquellos que aún no logró a la derecha. Por simplicidad, la Figura 5 muestra sólo tres de los alfas del núcleo.

**Determine a dónde ir con el núcleo.** Una vez el equipo se pone de acuerdo en los estados de los alfas, los miembros discuten los próximos estados “objetivo” deseados, que guían su planeación. El equipo se pone de acuerdo para usar los inmediatamente próximos estados de los alfas para ayudar a establecer los objetivos de la próxima iteración, como se muestra en la Figura 6.

El nombre del estado del alfa suministra una pista para comprender lo que se necesita lograr para alcanzar un estado. Los miembros del equipo pueden encontrar más al leer y comprender las listas de chequeo de los estados de los alfas. Al recorrer los estados de cada alfa uno a uno, un equipo se puede familiarizar rápidamente con lo que se requiere para lograr cada estado. De esta manera, el equipo aprende sobre los alfas del núcleo al mismo tiempo que determina el estado actual de su desarrollo y sus estados objetivo siguientes.

**Determine cómo llegar allá con el núcleo.** Smith y su equipo buscan los próximos estados objetivo y se ponen de acuerdo en lo que necesitan para establecer algunas prioridades. Primero, ellos necesitan determinar su forma de trabajo: trabajando bien, luego su sistema de software: usable y, finalmente, sus requisitos: tratados. La razón es simple: si no se tiene la forma de trabajo trabajando bien, se impedirían sus intentos de tener el sistema de software usable. Además, ellos se pusieron de acuerdo en la prioridad para los ítems de requisitos perdidos necesarios para lograr el estado requisitos: tratados.

Smith y su equipo luego discutieron lo que necesitaban hacer para lograr esos estados, como se muestra en la Tabla 1. Al recorrer los estados objetivo de los alfas, Smith es capaz de determinar un conjunto de objetivos y tareas para la próxima iteración.

Cómo el núcleo ayuda a planear iteraciones. Un buen plan debe ser inclusivo, lo que significa que debe incluir todos los ítems esenciales y cubrir el equipo como un todo. También, debe ser concreto, de modo que el equipo lo pueda accionar. El equipo debería, también, tener una manera de monitorear su progreso contra el plan.

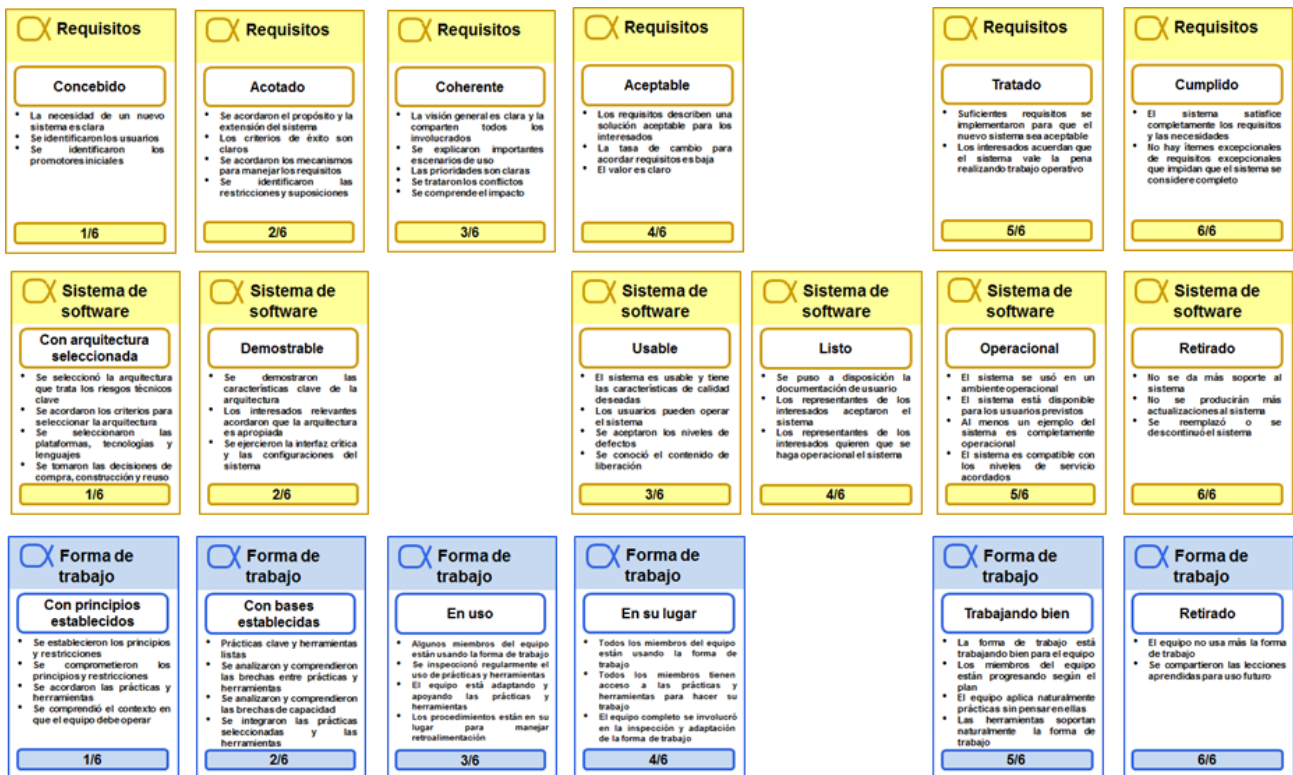


Fig. 5. El equipo usa los alfas para determinar los estados actuales.

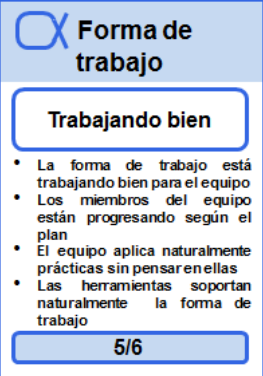
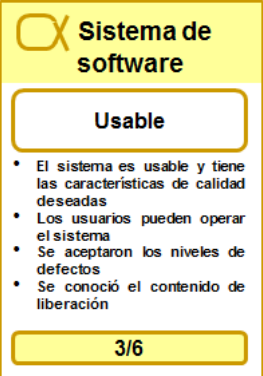
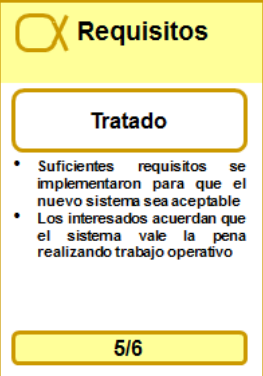
<b><i>Estado objetivo</i></b>	<b><i>Cómo el equipo planea alcanzarlo</i></b>
 <p><b>Forma de trabajo</b></p> <p><b>Trabajando bien</b></p> <ul style="list-style-type: none"> <li>• La forma de trabajo está trabajando bien para el equipo</li> <li>• Los miembros del equipo están progresando según el plan</li> <li>• El equipo aplica naturalmente prácticas sin pensar en ellas</li> <li>• Las herramientas soportan naturalmente la forma de trabajo</li> </ul> <p>5/6</p>	<p>Tanto Dick como Harriet estuvieron de acuerdo en que tenían dificultades para aplicar las pruebas automáticas. Ellos necesitaban ayuda para progresar. Tom estuvo de acuerdo en que él tenía que invertir tiempo enseñándoles.</p> <p>Se agregó una tarea a la lista de pendientes de la iteración para que Tom condujera un entrenamiento en pruebas automáticas para Dick y Harriet.</p> <ul style="list-style-type: none"> <li>• Tarea 1. Conducir entrenamiento en pruebas automáticas.</li> </ul>
 <p><b>Sistema de software</b></p> <p><b>Usable</b></p> <ul style="list-style-type: none"> <li>• El sistema es usable y tiene las características de calidad deseadas</li> <li>• Los usuarios pueden operar el sistema</li> <li>• Se aceptaron los niveles de defectos</li> <li>• Se conoció el contenido de liberación</li> </ul> <p>3/6</p>	<p>Este estado nos recuerda que el sistema de software debe mostrar que es de la calidad y funcionalidad suficientes para ser útil para los usuarios. Hasta acá, el equipo de Smith estuvo haciendo pruebas dentro de su entorno de desarrollo. Ahora, se deberían conducir pruebas dentro de un entorno de aceptación de pruebas, que debían preparar. Esto generó las siguientes tareas:</p> <ul style="list-style-type: none"> <li>• Tarea 2: Preparar el entorno de pruebas de aceptación. El equipo de Smith tenía que lograr que todos los ítems de requisitos fueran actualmente demostrables en el sistema para completarlos. Esto significaba que cada ítem de requisitos se debía probar completamente dentro del entorno de aceptación de pruebas.</li> <li>• Tarea 3: Completar el ítem de requisitos A: “Navegar en línea y fuera de línea.</li> <li>• Tarea 4: Completar el ítem de requisitos B: “Publicar comentarios (en línea y fuera de línea).</li> <li>• Tarea 5: Completar el ítem de requisitos C: “Navegar álbumes”.</li> </ul>
 <p><b>Requisitos</b></p> <p><b>Tratado</b></p> <ul style="list-style-type: none"> <li>• Suficientes requisitos se implementaron para que el nuevo sistema sea aceptable</li> <li>• Los interesados acuerdan que el sistema vale la pena realizando trabajo operativo</li> </ul> <p>5/6</p>	<p>Este estado nos recuerda la necesidad de trabajar con los interesados para asegurar que ellos están felices con el sistema producido. En nuestra historia, Smith tenía que trabajar con Angela, la representante del cliente, para determinar cuáles ítems de requisitos adicionales se necesitaba implementar. Esto generó la siguiente tarea adicional:</p> <ul style="list-style-type: none"> <li>• Tarea 6: Hablar con Angela y acordar con ella los ítems de requisitos, dentro de la iteración, que harán que el sistema sea digno de considerarse operativo.</li> </ul>

Fig. 6. Los próximos pasos seleccionados y la manera como los alcanzará el equipo.

El núcleo le ayuda a lograr esto de la manera siguiente:

*Inclusivo.* Los alfas del núcleo sirven como recordatorios de las diferentes dimensiones del desarrollo de software, ayudando a crear un plan que trate todas las dimensiones de manera balanceada.

*Concreto.* Las listas de chequeo para cada estado de alfa suministran pistas sobre lo que se necesita hacer en la iteración. Las listas de chequeo mismas ayudan a determinar su progreso, haciendo claro lo que usted tiene que hacer y comparando esto con lo que se supone que usted hace.

## V. EL NÚCLEO EN EL MUNDO REAL

Si bien las ideas que aquí se presentan serán nuevas para muchos de ustedes, la academia y la industria las vienen aplicando de forma exitosa en el mundo real. En todos los

casos, ellos usan el núcleo y las prácticas que se desarrollaron en Ivar Jacobson International [1, 10]. Entre los primeros en adoptar la idea del núcleo se cuentan:

- MunichRe, la compañía líder en seguros del mundo, donde una familia de “modelos de colaboración” se ensambló para cubrir el espectro completo del software y el trabajo de aplicación. Cuatro modelos de colaboración (exploratorio, estándar, mantenimiento y soporte) se construyeron en el núcleo mismo desde el mismo conjunto de doce prácticas.
- Fujitsu Services, donde el paquete Apt se construyó sobre una versión previa del núcleo de la ingeniería de software, incluyendo formas de trabajo ágiles y en cascada [1].
- Una de las mayores compañías japonesas de electrónica de consumo, donde los procesos de software se definieron sobre una versión previa del núcleo, ayudando a los

equipos a aplicar nuevas prácticas y a gestionar un proveedor de desarrollo externo.

- KPN, donde un proceso basado en el núcleo se adaptó para más de 300 proyectos a lo largo de trece programas como parte de un movimiento hacia el desarrollo iterativo. El núcleo también suministró la base para un nuevo proceso de aseguramiento de la calidad enfocado en los resultados, que se podría aplicar a todos los proyectos sin importar el método o las prácticas que se empleen.
- El mayor departamento de gobierno del Reino Unido, donde un paquete ágil basado en el núcleo se introdujo para habilitar la agilidad disciplinada y el seguimiento del progreso y salud de los proyectos de forma independiente a las prácticas.

El núcleo ya se usa en cursos de ingeniería de software de primero y segundo año en el Instituto Real de Tecnología KTH en Suecia. Después de que los estudiantes de los cursos de primer año dirigieron sus proyectos, se adentraron en los alfás de Semat y compararon los resultados de sus proyectos, bajo la dirección de Anders Sjögren. Los estudiantes tuvieron la oportunidad de familiarizarse con los alfás y evaluarlos para obtener una visión del progreso y salud de los proyectos. En los cursos de segundo año que condujo Mira Kajko-Mattsson, a los estudiantes se les preguntó el uso del núcleo de Semat cuando corrían sus proyectos sin importar el método que siguieran. Como se muestra en la Figura 3, Kajko-Mattsson creó un escenario de desarrollo de software y lo evaluó para cada alfa, sus estados y los ítems de la lista de chequeo de estados. A los estudiantes se les pidió, luego, hacer lo mismo cuando dirigían y evaluaban sus proyectos.

Las experiencias de esos cursos proporcionaron lecciones muy valiosas. Por ejemplo, el núcleo asegura que todos los aspectos esenciales de la ingeniería de software se consideran en un proyecto. Al emparejar los resultados contra los alfás del núcleo, los estudiantes pudieron identificar fácilmente los lados bueno y malo de sus métodos de desarrollo. El núcleo también preparó a los estudiantes para los futuros esfuerzos de desarrollo de software con un mínimo de esfuerzo en enseñanza. Siguiendo los alfás del núcleo, los estudiantes pudieron aprender el alcance total del esfuerzo en ingeniería de software y, así, vieron lo que se requería de ellos en su futuro como profesionales (véase la Figura 7).

**Cómo se relaciona el núcleo con lo ágil y otros paradigmas.** El núcleo se puede usar con todas las prácticas técnicas y administrativas populares, incluyendo Scrum, Kanban, iteraciones basadas en riesgos, cascada, desarrollo dirigido por casos de uso, desarrollo dirigido por pruebas de aceptación, integración continua y desarrollo dirigido por pruebas. Esto ayudará a los grupos a embarcarse en el desarrollo de productos de software nuevos e innovadores, y aquellos involucrados en mejorar y mantener productos de software establecidos. Esto ayudará, también, a equipos de todos los tamaños, desde equipos individuales hasta programas fuertes de ingeniería de software de más de 1000 personas.

Por ejemplo, el núcleo apoya los valores del Manifiesto Ágil. Con su enfoque en listas de chequeo y resultados y su inherente independencia de las prácticas, el núcleo valora los individuos y las interacciones por encima de los procesos y las herramientas. Con su enfoque en las necesidades de los equipos profesionales de desarrollo de software, el núcleo valora la forma de trabajo y privilegia las responsabilidades del equipo por encima de los métodos.

El núcleo no compite de ninguna manera con los métodos existentes, sean ellos ágiles o de otro tipo. Por el contrario, el núcleo es agnóstico para el método que elija el equipo. Aún si su equipo está usando un método particular, el núcleo aún puede ayudar. Sin importar el método que se use, como Robert Martin señala en su prefacio a *La esencia de la ingeniería de software*, los proyectos, aún los ágiles, se pueden desordenar y, cuando lo hacen, los equipos necesitan saber más. Ahí es donde yace el verdadero valor del núcleo, pues puede guiar un equipo a las acciones que necesitan realizar para regresar a la senda, para extender su método o para tratar un vacío crítico en su forma de trabajo. Se enfoca en las necesidades del profesional de software y los valores del “uso de los métodos” por encima de “la descripción de las definiciones de los métodos” (como era normal en el pasado).

El núcleo no sólo apoya las mejores prácticas modernas, sino que también reconoce que una gran cantidad de software ya se desarrolló y necesita que lo mantengan. Este software vivirá por décadas y lo tendrán que mantener de manera eficiente. Esto significa que la forma como trabaje este software tendrá que evolucionar a lo largo del software mismo.

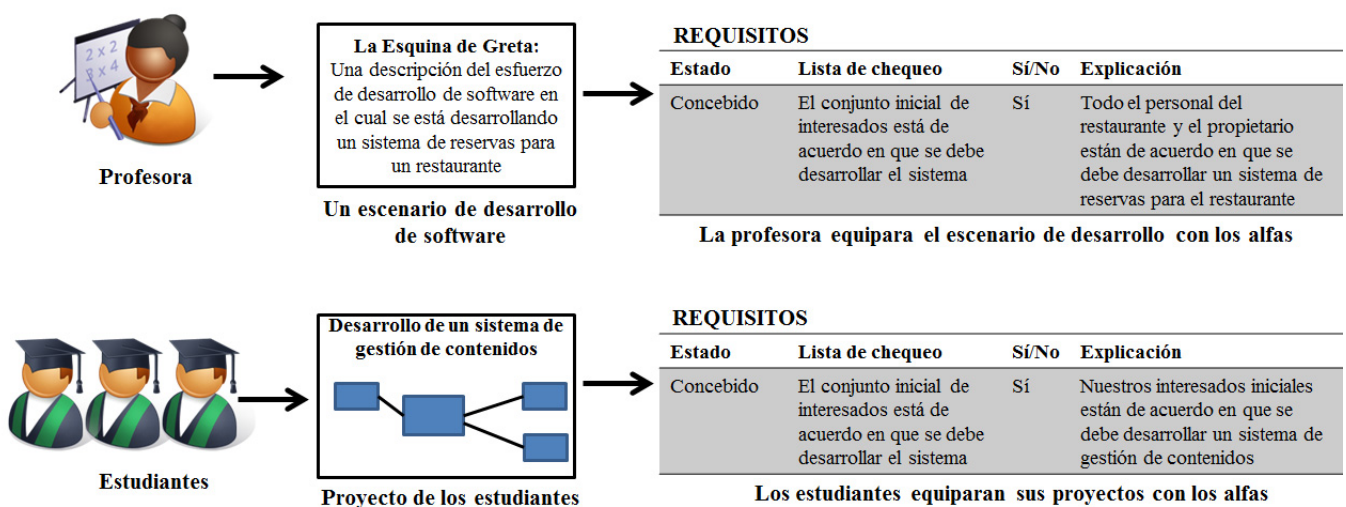


Fig. 7. Visualización del enfoque pedagógico cuando se enseña con el núcleo de Semat.



Nuevas prácticas se tendrán que introducir de manera que complementen las que ya están en uso. El núcleo proporciona los mecanismos para migrar los métodos legados desde los monolíticos enfoques en cascada hasta los más modernos métodos ágiles y más allá, de forma evolucionaria. Esto le permite cambiar sus métodos legados práctica a práctica, mientras mantiene y mejora la habilidad del equipo para hacer entregas.

**Cómo le ayuda el núcleo.** El uso del núcleo tiene muchos beneficios para los profesionales de software, ya sean experimentados o novatos, y para los equipos en que trabajan. Por ejemplo, el núcleo ayuda a evaluar el progreso y salud de los esfuerzos de desarrollo de software, evaluar las prácticas actuales y mejorar la forma de trabajo. También, ayuda a mejorar la comunicación, a moverse más fácilmente entre equipos y a adoptar nuevas ideas. El núcleo ayudará a la industria en conjunto, al mejorar la interoperabilidad entre los equipos, los proveedores y las organizaciones de desarrollo.

Al proporcionar un fundamento independiente de las prácticas para la definición de los métodos de software, el núcleo tiene también el poder de transformar completamente la forma en que se definen los métodos y se intercambian las prácticas. Por ejemplo, al permitir a los equipos mezclar y emparejar prácticas de diferentes fuentes para construir y mejorar la forma de trabajo, el núcleo trata dos de los problemas metodológicos clave que enfrenta la industria:

- Los equipos no se encuentran ya atrapados en sus métodos. Ellos pueden mejorar continuamente su forma de trabajo al agregar o remover prácticas cuando la situación lo exija.
- Los metodólogos no necesitan más gastar tiempo en describir los métodos completos. Ellos pueden fácilmente describir sus nuevas ideas de manera concisa y reutilizable.

Finalmente, el núcleo beneficia la academia. El núcleo proporciona la base para la creación de cursos fundamentales de la ingeniería de software, que se pueden luego complementar con cursos adicionales sobre prácticas específicas (ya sea parte del currículum educativo inicial o después durante el desarrollo profesional de los estudiantes). Igualmente importante es la habilidad del núcleo para actuar como un modelo de referencia compartido y habilitar la investigación y experimentación adicional.

#### REFERENCIAS

[1] Azoff, M. Apt Methods and Tools, Fujitsu. Ovum Technology Report. Reference Code O100032-002 (Jan. 2011).

[2] Fujitsu, Ivar Jacobson International AB, Model Driven Solutions. Essence—kernel and language for software engineering. Initial submission, 2012, version 1.0.

[3] Jacobson, I. and Meyer, B. Methods need theory. Dr. Dobb's Journal, (2009).

[4] Jacobson, I., Meyer, B. and Soley, R. Call for action: The SEMAT initiative. Dr. Dobb's Journal, (2009).

[5] Jacobson, I., Meyer, B. and Soley, R. The SEMAT vision statement, (2009).

[6] Jacobson, I., Pan-Wei Ng, McMahon, P., Spence, I. and Lidman S. The Essence of Software Engineering—Applying the SEMAT Kernel. Addison-Wesley. (Forthcoming in Jan. 2013 but available in a prepublication version on safaribooksonline.com.)

[7] Jacobson, I., Pan-Wei Ng and Spence, I. Enough of process—let's do practices. Journal of Object Technology 6, 6 (2007), 41-67.

[8] Jacobson, I. and Spence, I. Why we need a theory for software engineering. Dr. Dobb's Journal, (2009).

[9] Object Management Group (OMG). RFP: A foundation for the Agile creation and enactment of software engineering methods, (2012).

[10] Pan-Wei Ng and Magee, M. Lightweight application lifecycle management using state cards. Agile Journal (Oct. 2010)



**Ivar Jacobson**, el presidente de Ivar Jacobson International, es uno de los padres de los componentes y de la arquitectura de componentes, los casos de uso, el lenguaje unificado de modelado y el proceso unificado racional. Ha contribuido al modelado de negocios moderno y al desarrollo de software orientado a aspectos.



**Pan-Wei Ng** entrena el desarrollo de sistemas a gran escala que involucran muchos millones de líneas de código y cientos de personas por cada versión, ayudándoles a la transición hacia una forma de trabajo lean y ágil, sin olvidar el mejoramiento de su código y arquitectura y las pruebas por medio de casos de uso y aspectos. Es coautor, con Ivar Jacobson, del libro “Desarrollo de software orientado a aspectos con casos de uso”.



**Paul McMahon** es un consultor independiente que se enfoca en el entrenamiento de gerentes de proyectos, líderes de proyectos y profesionales de software en el uso práctico de las técnicas lean y ágiles en entornos restringidos. Ha sido líder de la iniciativa Semat desde sus inicios.



**Ian Spence** es oficial de tecnología en jefe en Ivar Jacobson International y el líder del equipo para el desarrollo del núcleo de Semat. Introdujo cientos de proyectos a las prácticas iterativas y ágiles y también lideró numerosos proyectos exitosos de transformación a gran escala trabajando con organizaciones de más de 5000 personas.



**Svante Lidman** tiene una amplia experiencia en la construcción de equipos de desarrollo de software en empresas de alto rendimiento. Ha ocupado cargos en Hansoft AB, Ivar Jacobson International, Microsoft, Rational Software y Objectory, entre otras. Desde mediados de 2010, Lidman es el agente de cambio que lidera la más grande transición lean/ágil de todos los tiempos en Escandinavia.



**Carlos Mario Zapata** se desempeña actualmente como Profesor Asociado del Departamento de Ciencias de la Computación y de la Decisión de la Universidad Nacional de Colombia, Sede Medellín. Es, además, presidente del Comité Ejecutivo del Capítulo Latinoamericano de Semat y también es uno de los traductores oficiales del libro “La Esencia de la Ingeniería de Software: aplicando el núcleo de Semat”.