## [Principios](javascript:void(0))

La comunidad Lean Software & Systems parece estar de acuerdo con algunos principios que respaldan los procesos de Lean Software Development.

* Siga Systems Thinking & Design Approach
* Los resultados emergentes pueden estar influenciados por la creación de la arquitectura del contexto para un sistema adaptable complejo
* Respetar a las personas (como parte del sistema)
* Utilice el modo científico (para controlar las mejoras)
* Fomentar el liderazgo
* Genere visibilidad (en los trabajos, flujos de trabajo y operación del sistema)
* Reducir el tiempo de flujo
* Reducir residuos para mejorar la eficacia

### [Siga Systems Thinking & Design Approach](javascript:void(0))

Esto se conoce a menudo en la literatura de Lean como “optimizar el todo”, lo que implica que es el resultado de todo el sistema (o el proceso) lo que se desea optimizar, y no se deben optimizar partes con la esperanza equivocada de que esto optimizará el conjunto mágicamente. La mayoría de los usuarios creen que la consecuencia es así, que la optimización de elementos (optimización local) dará lugar a un resultado que no será óptimo.

Lean Systems Thinking and Design Approach requiere tener en cuenta las peticiones en el sistema realizadas por las partes interesadas externas, como clientes, y el resultado deseado requerido por tales partes interesadas. Debemos estudiar la naturaleza de la demanda y compararla con la capacidad de entrega de nuestro sistema. La petición incluirá la denominada “petición de valor”, por la que los clientes están dispuestos a pagar, y la “petición de errores”, que normalmente implica una revisión o una petición adicional a causa de un error en el suministro de la de petición de valor. Las peticiones erróneas tiene dos formas por lo general: revisión del trabajo respecto a una petición de valor previamente satisfecha y servicios adicionales o soporte debido a un error a la hora de entregar la petición de valor. En el desarrollo de software, la demanda de error suele solicitar una corrección de errores así como una función de atención al cliente o de soporte técnico.

Un enfoque de diseño de sistemas requiere que también sigamos el enfoque de planear, hacer, estudiar, actuar (Plan-Do-Study-Act, PDSA) para procesar el diseño y las mejoras. Hora Edwards Deming usó las palabras “estudiar” y “capacidad” para indicar que estudiamos la filosofía natural del comportamiento del sistema. Este sistema se compone de nuestro proceso de desarrollo de software y de todas las personas involucradas en él. Tendrá un comportamiento visible en términos de plazos de entrega, calidad, cantidad de las características o funciones entregadas (lo cual se denominado en la literatura de Agile “velocidad "), etc. Estas métricas mostrarán variabilidad y, estudiando la media y la dispersión de las variaciones, podemos entender mejor nuestra capacidad. Si no se corresponden correctamente con la demanda y las expectativas del cliente, el sistema necesitará ser rediseñado para cerrar la brecha.

Deming también nos enseñó que la capacidad está influenciada en un 95% por el diseño del sistema y solo el 5% está condicionado por el rendimiento de las personas. Es decir, podemos respetar a personas no culpándolas por un desequilibrio entre capacidad y la demanda y rediseñando el sistema para permitirles hacer bien las cosas.

Para entender diseño de sistemas, debemos tener conocimientos científicos de la dinámica de la capacidad del sistema y cómo puede verse afectada. Los modelos se desarrollan para predecir la dinámica del sistema. Cuando hay muchos modelos posibles, se suelen usar los más populares: la comprensión de costos económicos; los denominados costos de transacción y de coordinación relacionados con la producción de productos o servicios valorados por el cliente; la teoría de las restricciones (la comprensión de los cuellos de botella); y la teoría del conocimiento profundo (el estudio y el reconocimiento de si las variaciones son propias del diseño del sistema, o especiales y externas al diseño del sistema).

### [Los resultados emergentes pueden estar influenciados por la creación de la arquitectura del contexto para un sistema adaptable complejo](javascript:void(0))

Los sistemas complejos tienen condiciones de inicio y reglas simples que, cuando se ejecutan en iteración, generan un resultado emergente. Los resultados emergentes son difíciles o imposibles de predecir, dadas las condiciones de inicio. El experimento informático “The Game of Life” es un ejemplo de un sistema complejo. Un sistema adaptable complejo incluye cierta conciencia y un método interno de reflexión que le permita tener en cuenta en qué medida su conjunto de reglas actual le permite lograr los resultados deseados. El sistema adaptable complejo podrá entonces optar por adaptarse él mismo (para cambiar sus reglas simples) para cerrar el hueco entre el resultado actual y el resultado deseado. El juego de la vida adaptado de modo que las reglas se puedan volver a escribir durante el juego sería un sistema adaptable complejo.

En los procesos de desarrollo de software, las “reglas sencillas” de sistemas adaptables complejos son las directivas que constituyen la definición de proceso. El principio básico aquí se basa en la creencia de que desarrollar productos y servicios de software no es una actividad determinista, y por consiguiente un proceso definido que no puede adaptarse a sí mismo no será una respuesta adecuada a eventos imprevistos. Por consiguiente, el proceso diseñado como parte de nuestro patrón de pensamiento y enfoque del diseño debe ser personalizable. Se adapta a través de la modificación de las directivas de las que se compone.

El enfoque de Kanban a Lean Software Development utiliza este concepto tratando las directivas del sistema de extracción kanban como “las reglas sencillas”, y las condiciones iniciales son que el trabajo y el flujo de trabajo se visualizan, el flujo se administra mediante la comprensión del la dinámica del sistema y la organización utiliza un enfoque científico para comprender, proponer e implementar mejoras de procesos.

### [Respetar a las personas](javascript:void(0))

La comunidad Lean adopta la definición de Peter Drucker de trabajo de conocimiento que afirma que los trabajadores son trabajadores de conocimiento si están más informados sobre el trabajo que realizan que sus jefes. Esto crea la implicación de que los trabajadores están en una posición mejor para tomar decisiones sobre cómo realizar el trabajo y cómo modificar los procesos para mejorar los métodos de trabajo. Por lo que la voz del trabajador debe ser respetada. Los trabajadores deben estar facultados para organizarse por su cuenta para completar el trabajo y lograr los resultados deseados. También deben tener autorización para sugerir e implementar oportunidades de mejora de procesos o “eventos de kaizen”, como se conocen en la literatura Lean. Crear directivas de proceso explícitas de modo que los trabajadores sean conscientes de reglas que restringen su actividad es una forma de respetarles. Las reglas bien definidas fomentan una organización autónoma ya que eliminan el miedo y la necesidad de tener valor. Respetar a las personas dándoles autoridad y ofreciéndolas un conjunto de directivas declaradas explícitamente permite ser fiel al valor básico de respeto a la condición humana.

### [Use el método científico](javascript:void(0))

Utilice modelos para entender la dinámica de cómo se realiza el trabajo y de cómo funciona el sistema Lean Software Development. Observe y estudie el sistema y su capacidad y, a continuación, desarrolle y aplique los modelos para predecir su comportamiento. Recopile datos cuantitativos en las revisiones y use esos datos para entender cómo rinde el sistema y predecir cómo puede cambiar cuando cambia el proceso.

La comunidad Lean Software & Systems utiliza métodos estadísticos, como gráficos de control de proceso estadístico e histogramas de análisis espectral de datos sin formato para conseguir tiempo y velocidad para comprender la capacidad del sistema. También utilizan modelos como: la teoría de las restricciones para entender los cuellos de botella; el sistema de conocimiento profundo para entender la variación inherente al diseño del sistema frente a la que procede de influencias externas; y un análisis de costos económicos para las tareas meramente de coordinación, configuración, entrega o limpieza posteriores a la creación de productos o servicios valorados por el cliente. Otros modelos se están empezando a utilizar, como la teoría de la opción de real, que intenta aplicar teoría de la opción financiera de la administración de riesgos financieros a la toma de decisiones del mundo real.

El método científico sugiere: estudiamos; postulamos un resultado en función de un modelo; perturbamos el sistema basándonos en esa predicción; y observamos de nuevo para ver si la perturbación produjo los resultados que el modelo predijo. De no ser así, comprobamos los datos y reconsideramos si nuestro modelo es exacto. El uso de modelos para controlar las mejoras de procesos lo convierte en una actividad científica, dejando de ser una actividad supersticiosa basada en la intuición.

### [Fomentar el liderazgo](javascript:void(0))

La dirección y administración no son iguales. Administración es la actividad de diseñar procesos, crear, modificar y eliminar directivas, de tomar decisiones estratégicas y operativas, de recopilar recursos, de proporcionar financiación e instalaciones y de comunicar información sobre contexto como estrategia, objetivos y resultados deseados. La dirección tiene que ver con la visión, la estrategia, la táctica, el valor, la innovación, la valoración, la propugnación y muchos otros atributos. La dirección puede y debe proceder de cualquier persona dentro de una organización. Pequeños actos de liderazgo de los trabajadores crearán una cascada de mejoras que posibilitarán los cambios necesarios para crear un proceso de Lean Software Development.

### [Genere visibilidad](javascript:void(0))

El trabajo de conocimiento no es visible. Si no puede ver algo, es (casi) imposible administrarlo. Es necesario generar visibilidad en el trabajo que se emprende y en el flujo de ese trabajo a través de una red de personas, conocimientos y departamento hasta que se complete. Es necesario crear visibilidad en el diseño del proceso encontrando maneras de visualizar el flujo del proceso y creando directivas del proceso explícitas para que todo el mundo las vea y las tenga en cuenta. Cuando todas estas cosas estén visibles, el uso del método científico es posible, y las conversaciones sobre posibles mejoras podrán ser colaborativas y objetivas. La mejora del proceso de colaboración es casi imposible si el trabajo y el flujo de trabajo no están visibles y si las directivas de proceso no son explícitas.

### [Reducir el tiempo de flujo](javascript:void(0))

Los profesionales del desarrollo de software y el personal académico que estudia ingeniería de software tradicionalmente se han centrado en medir el tiempo empleado en trabajar en una actividad. La comunidad de Lean Software Development ha descubierto que podría ser más útil medir el tiempo transcurrido real que algo tarda en procesarse. Esto se conoce normalmente como duración del ciclo y normalmente se califica mediante los límites de las actividades realizadas. Por ejemplo, *la duración del ciclo hasta llegar a Listo para la implementación mediría el tiempo total transcurrido para un elemento de trabajo, como un caso de usuario, que debe analizarse, diseñarse, desarrollarse, probarse en varios sentidos, y colocarse en cola listo para su implementación en un entorno de producción.*

El centrarse en el tiempo que se necesita para recorrer el proceso es importante de varias maneras. Se *han mostrado duraciones de ciclo más largas que se correlacionan con un crecimiento no lineal en el porcentaje de errores. Por consiguiente, los tiempos del ciclo llevan a una mayor calidad*. Esto es contraintuitivo, ya que parece ridículo que los errores se puedan insertar en el código mientras está esperando en la cola y realmente no lo está tocando nadie. Tradicionalmente, la profesión de ingeniería del software y los académicos que la han estudiado han omitido este tiempo de inactividad. Sin embargo, *las pruebas empírica sugieren que la duración de ciclo es importante para la calidad inicial.*

Alan Shalloway también habla sobre el concepto de “trabajo inducido”. Su observación es que un retraso en realizar una tarea puede conducir a que esa tarea que requiera muchos más esfuerzos de los necesarios. Por ejemplo, encontrar y corregir inmediatamente un error puede llevar tan solo 20 minutos, pero si ese error se evalúa, se pone en la cola y después se esperan varios días o semanas a que se solucione, podría llevar muchas horas su corrección. Por consiguiente, el retraso en el tiempo del ciclo tiene trabajo adicional “inducido”. Como este trabajo es evitable, en términos de Lean, debe considerarse como un “desperdicio”.

La tercera razón para centrarse en la duración del ciclo es una *razón relacionada con el negocio.* Cada característica, función o caso de usuario tiene un valor. Ese valor puede ser incierto pero, sin embargo, hay un valor. El valor puede variar con el tiempo. El concepto de valor variable con el tiempo se puede expresar económicamente como una función de rentabilidad de mercado. Cuando se comprende la función de rentabilidad de mercado para un elemento de trabajo, aunque la función presente una variedad de valores para modelar la incertidumbre, es posible evaluar un “costo de retraso”. El costo de retraso permite poner un valor en la reducción del tiempo del ciclo.

Con algunos elementos de trabajo, la función de rentabilidad de mercado no se inicia hasta una fecha futura conocida. Por ejemplo, una característica diseñada para utilizarse durante el día festivo del 4 de julio en los Estados Unidos de América no tiene valor antes de esa fecha. Acortar el tiempo de ciclo y poder predecir su duración con cierta confiabilidad sigue siendo útil en este ejemplo. En el mejor de los casos, queremos empezar con el trabajo para entregar la característica “en el plazo exacto” cuando sea necesaria y no significativamente antes de la fecha deseada, ni tarde, ya que una entrega fuera del plazo conlleva costos por retraso. La entrega a su debido tiempo garantiza que el uso óptimo se creó con recursos disponibles. Una entrega antes de tiempo implica que puede que hayamos trabajado en algo más y, implícitamente, haber incurrido en un costo de oportunidad por aplazamiento.

Como resultado de estas tres razones, Lean Software Development *busca minimizar el tiempo de flujo y grabar los datos que permitan realizar predicciones sobre el tiempo de flujo. El objetivo es minimizar la demanda por errores debida a errores, los residuos de la sobrecarga debida al retraso en la solución de errores y maximizar el valor evitando los costos por retraso y los costos de oportunidad por aplazamiento.*

### [Reducir residuos para mejorar la eficacia](javascript:void(0))

Para cada actividad que agregue valor, hay actividades de configuración, de limpieza y de entrega que son necesarias pero no agregan valor por sí mismas. Por ejemplo, una iteración del proyecto que desarrolla un incremento de software que funciona requiere un planeamiento (una actividad de configuración), un entorno y quizás una bifurcación de código en el control de versiones (conocido en su conjunto como administración de configuración y también como actividad de configuración), un plan de lanzamiento y realizar el lanzamiento en sí (una actividad de entrega), una demostración al cliente (una actividad de entrega) y posiblemente destruir o reconfigurar el entorno (una actividad de limpieza.) En términos económicos, las actividades de configuración, limpieza y entrega son costos de transacción para llevar a cabo el trabajo de valor añadido. Estos costos (o gastos generales) se consideran residuos en la metodología Lean.

Cualquier forma de sobrecarga de comunicación puede considerarse un desperdicio. Las reuniones para determinar el estado del proyecto y para programar o asignar trabajo a miembros del equipo se considerará un costo de coordinación en terminología económica. Todos los costos de coordinación son irrelevantes en esta percepción de Lean. *Los métodos de Lean Software Development pretenden eliminar o reducir los costos de coordinación con la colocación de miembros del equipo, reuniones cara a cara de poca duración como reuniones sin sentarse y controles visuales como los muros de cartas.*

El tercer tipo común de residuos del Lean Software Development es la demanda por errores. Las peticiones erróneas son una carga para sistema de desarrollo de software. Las peticiones de erróneas suelen conllevar un trabajo de revisión u otras formas de trabajo como efecto secundario de una baja calidad. Las formas más habituales de demanda por errores en el desarrollo de software son los errores, los defectos de producción y las actividades de soporte al cliente derivadas de un uso incorrecto del software. El porcentaje de trabajo en curso que es demanda por errores suele denominarse carga por errores. El porcentaje de trabajo de valor añadido frente a la demanda por errores es una medida de la eficacia del sistema.

El porcentaje de trabajo de valor añadido frente al trabajo total, incluidos todos los costos de coordinación y transacción que no añaden valor, determina el nivel de eficacia. Un sistema sin costes de transacción y de coordinación y las cargas erróneas no se considerarán eficaces al 100%.

Tradicionalmente, las ciencias empresariales occidentales han enseñado que la eficacia se puede mejorar aumentando el tamaño de lote de trabajo. Normalmente, los costos de transacción y coordinación son fijos o suben ligeramente con un aumento del tamaño de lote. Como resultado, los lotes grandes de trabajo son más eficaces. Este concepto se conoce como “economía de escala”. Sin embargo, con los problemas relacionados con los trabajos de conocimiento, los costos de coordinación tienden a subir de forma no lineal con el tamaño del lote, mientras que los costos de transacción puede presentar a menudo un crecimiento lineal. Como resultado, el enfoque tradicional del siglo XX relativo a la eficacia no es adecuado para los problemas de trabajo sobre los conocimientos el desarrollo de software.

Es mejor centrarse en la reducción de las sobrecargas conservando un tamaño pequeño de tamaño de lote para mejorar la eficacia. Por consiguiente, la forma según Lean de ser eficaz es reducir los desechos. Los métodos de Lean Software Development se centran en métodos rápidos, baratos y de planeación rápida; sobrecarga baja de comunicación; y mecanismos de coordinación de sobrecargas bajas efectivos, como controles visuales en sistemas kanban. También promueven la automatización de las pruebas y de la implementación para reducir los costos de transacción de las entregas. Las herramientas modernas para minimizar los costos de configuración y de destrucción de entorno, como sistemas de control de versiones modernos y uso de la virtualización, también ayudan a mejorar la eficacia de pequeños lotes de desarrollo de software.