Ingeniería de software I

METRICAS Y ESTIMACION DE PROYECTOS DE SOFTWARE

Fuente: material de Ing. Mario L. Peralta

Construir software de computadora es una **labor compleja**, particularmente si involucra a **muchas personas** que trabajan durante un tiempo relativamente largo. Por eso es necesario administrar los proyectos de software.

En cuanto inician las actividades de administración, se produce un ***plan de proyecto*** que define el ***proceso y las tareas*** que se van a realizar, el ***personal*** que hará el trabajo y los **mecanismos** que se emplearán para **valorar riesgos, controlar el cambio** y **evaluar** la **calidad**.

La ***administración efectiva*** de un proyecto de software se enfoca en las cuatro P:

personal,

producto,

proceso y

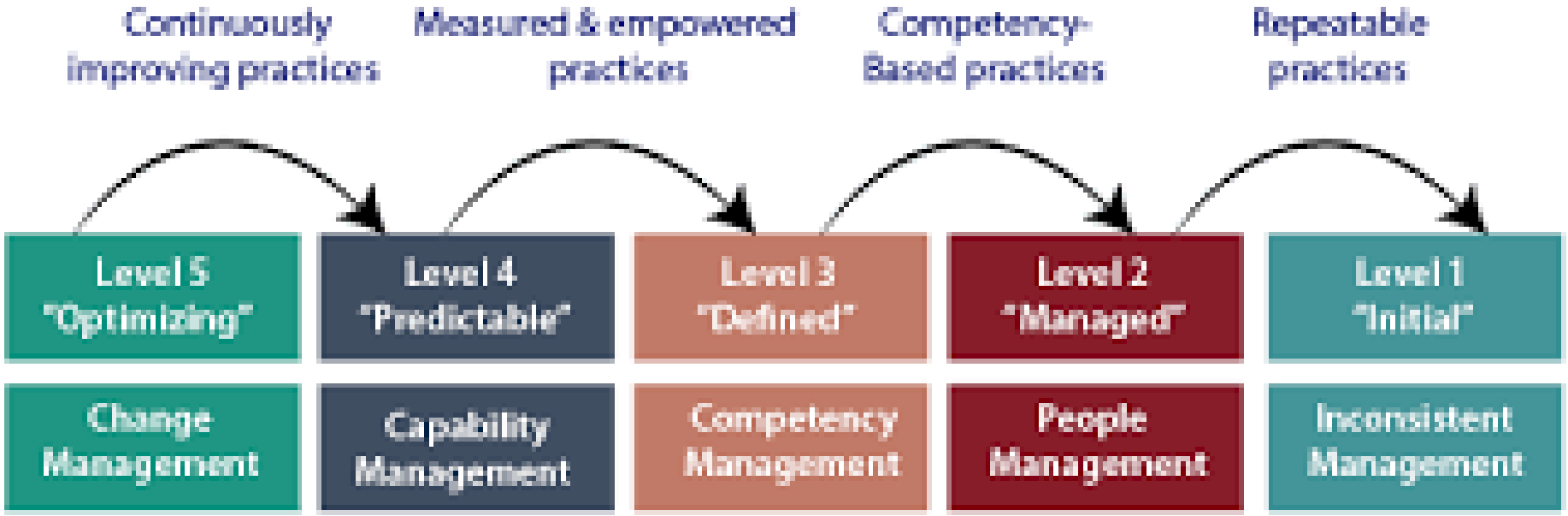
proyecto.

Personal

Personal.

Los administradores argumentan que las personas son lo importante, pero sus acciones en ocasiones contradicen sus palabras.

People CMM Maturity Levels



1. *Gerentes ejecutivos*, quienes definen los **temas empresariales** que con frecuencia tienen una influencia significativa sobre el proyecto.
2. *Gerentes de proyecto (técnicos)*, quienes deben **planificar**, motivar, **organizar** y **controlar** a los profesionales que hacen el trabajo de software.
3. *Profesionales* que aportan las **habilidades técnicas** que se necesitan para someter a ingeniería un producto o aplicación.
4. *Clientes* que **especifican los requerimientos** para el software que se va a fabricar, así como otros participantes que tienen un interés periférico en el resultado.
5. *Usuarios finales*, quienes **interactúan con el software** una vez que se libera para su uso productivo.

Para ser efectivo, el equipo de software debe organizarse de manera que ***maximice*** las ***habilidades*** y ***capacidades*** de cada persona.

Labor del **líder del equipo:**

Motivación

Organización

Ideas o innovación

Un gerente de proyecto de software debe concentrarse en comprender el problema que se va a resolver, en administrar el flujo de ideas y, al mismo tiempo, en dejar que todos en el equipo sepan (por medio de palabras y, mucho más importante, con acciones) que la calidad cuenta y que no se comprometerá

Para lograr un ***equipo de alto rendimiento:***

- Los miembros del equipo deben tenerse ***confianza*** entre sí.
- La distribución de ***habilidades*** debe ser adecuada para el problema.
- Es posible que tenga que ***excluirse*** del equipo a los ***inconformes*** si debe mantenerse la cohesión del equipo.

Sin importar el tipo de organización del equipo, el objetivo para todo gerente de proyecto es ayudar a crear un equipo que muestre cohesión

Equipos ágiles

La filosofía ágil alienta la ***satisfacción del cliente*** y la ***entrega incremental temprana*** del software.

Pequeños equipos de proyecto enormemente ***motivados***, métodos ***informales***, mínimos productos operativos de ingeniería de software y ***simplicidad*** de desarrollo global.

Equipos ágiles

Para hacer uso efectivo de las competencias de cada miembro del equipo y fomentar la colaboración efectiva a través de un proyecto de software, ***los equipos ágiles son autoorganizados***.

Producto

Producto.

Antes de poder planear un proyecto, deben establecerse los ***objetivos*** y el ***ámbito*** del ***producto***, considerarse soluciones alternativas e identificar las restricciones técnicas y administrativas.

Sino, es imposible definir ***estimaciones*** razonables del costo, una valoración efectiva del ***riesgo***, una descomposición realista de las ***tareas del proyecto*** y un ***calendario*** de proyecto manejable que proporcione en cada momento un indicio significativo del progreso

Producto.

Se requieren ***estimaciones cuantitativas*** y un plan organizado, pero no hay información sólida disponible. Un análisis detallado de los requerimientos del software proporcionaría la información necesaria para las estimaciones, pero el análisis usualmente tarda semanas o incluso meses en completarse

Ámbito del software,

Contexto.

Objetivos de información.

Función y desempeño.

Proceso

Proceso.

Un proceso de software proporciona el ***marco conceptual*** desde el cual puede establecerse un plan completo para el desarrollo de software.

> *"Quienes se adhieren a la filosofía de proceso ágil argumentan que su proceso es más esbelto que otros. Esto puede ser cierto, pero todavía tienen un proceso, y la ingeniería de software ágil todavía requiere disciplina."*
> *Pressman*

El problema es seleccionar el modelo de proceso que sea adecuado para el software que el equipo del proyecto someterá a ingeniería

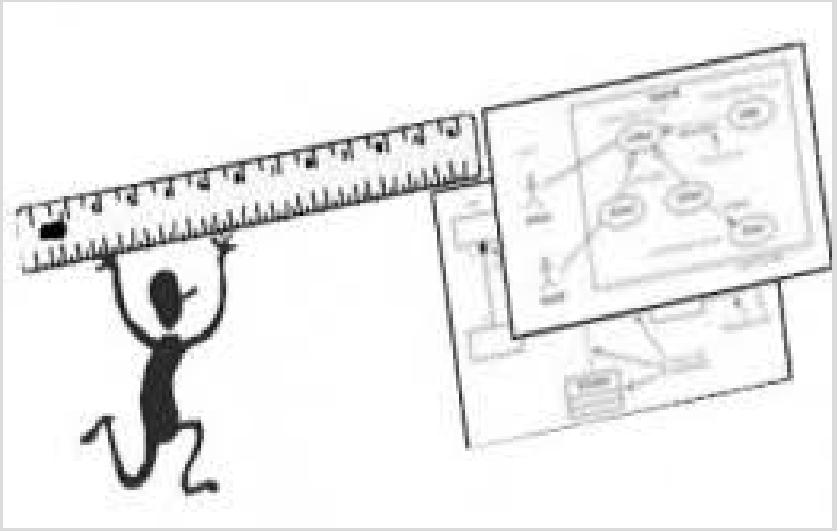
Proyecto

Señales que indican que un proyecto de sistemas de información está en **peligro**:

1. El personal del software no entiende las ***necesidades del cliente***.
2. El ámbito del producto está ***pobremente definido***.
3. Los ***cambios*** se gestionan ***pobremente***.
4. Cambia la ***tecnología*** elegida.
5. Las ***necesidades empresariales*** cambian o están mal definidas.
6. Las fechas límite son ***irreales***.
7. Los usuarios son ***resistentes***.
8. El equipo del proyecto carece de personal con ***habilidades*** adecuadas.
9. Los gerentes evitan mejores prácticas y lecciones aprendidas.

MÉTRICAS DE PROCESO Y DE PROYECTO

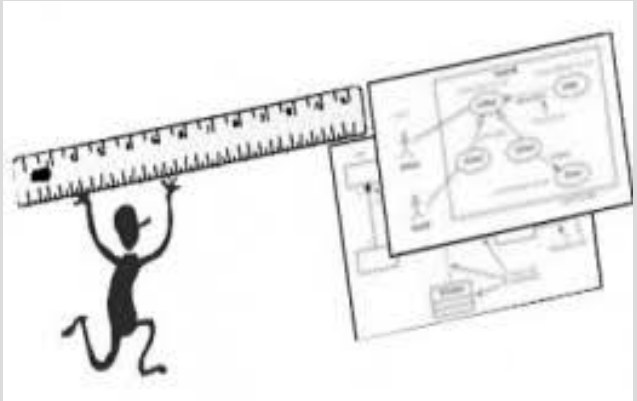
Las **métricas de proceso** se recopilan a través de todos los proyectos y durante largos espacios de tiempo. Su intención es proporcionar un ***conjunto de indicadores de proceso*** que conduzca a mejorar el proceso de software a largo plazo.



Las **métricas de proyecto** permiten

- valorar el estado de un proyecto en marcha,
- rastrear riesgos potenciales,
- descubrir áreas problema antes de que se vuelvan “críticas”,
- ajustar el flujo de trabajo o las tareas y
- evaluar la habilidad del equipo del proyecto para controlar la calidad de los productos operativos del software.

Las medidas que recopila un equipo de proyecto y que convierte en métricas para uso durante un proyecto también pueden transmitirse a quienes tienen responsabilidad en la mejora del proceso de software



La Gestión de Proyectos es un conjunto de actividades específicas que se emplean para la administración del proyecto. Estas actividades comprenden diversos aspectos:

- ***Estimación del esfuerzo*** necesario para el desarrollo de un Sistema de Información.
- Planificación de tareas y recursos.
- Control de tareas.
- Seguimiento del proyecto.
- Control de las incidencias.
- Control de cambios.

Estimación

La precisión de las estimaciones, condicionan el éxito del proyecto

Los parámetros involucrados en la estimación para proyectos de software son: **tamaño** del software, **esfuerzo** invertido, **tiempo** de desarrollo, **tecnología** utilizada, entre otros. La estimación de esfuerzo se mide en términos del esfuerzo requerido por **persona/mes,** normalmente expresado en términos de Horas/Hombre

La ***planificación*** es “la elaboración de un plan general, científicamente organizado y frecuentemente de gran amplitud, para obtener un objetivo determinado, tal como el desarrollo económico, la investigación científica, el funcionamiento de una industria”.

Importancia de la planificación de proyectos de software

- El software se ***desarrolla***, no se fabrica en un sentido clásico.
- Un proyecto comienza con la ***estimación del trabajo*** a realizar, estimación de recursos necesarios, tiempo y costo de desarrollo.
- La estimación conlleva un grado de ***incertidumbre***.
- Un buen planificador debe reducir esa incertidumbre.
- ***No existe modelo universal*** de estimación o fórmulas universalmente aplicables a todos los proyectos

El manejador de costo para un proyecto es sin duda el ***tamaño*** del proyecto. La ***medida del tamaño*** debe estar relacionada directamente con el ***esfuerzo de desarrollo***; por esta razón, las métricas de tamaño abarcan todos los aspectos que influyen en el costo, como tecnología, tipos de recursos y complejidad

Como soluciones para la disminución de incertidumbre en la estimación del tamaño del proyecto se considera el uso de ***registros históricos*** y esfuerzo requerido en proyectos anteriores

Paralelamente a las técnicas de modelado, han surgido diferentes técnicas de estimación para el desarrollo de software, de acuerdo a la evolución de los paradigmas de programación, tales como ***Líneas de Código Fuente (LOC), Puntos Función (PF), COCOMO I, COCOMO II***

Con el surgimiento del paradigma orientado a objetos y el modelado de Casos de Uso, se llevaron a cabo investigaciones sobre la factibilidad de implementar metodologías de estimación basadas en diagrama de clases de objetos y casos de uso. Una de estas investigaciones dio origen al **método de Puntos de Casos de Uso** (UCPs) creado por Gustav Karner. El método toma ideas de los Puntos Función (PF)

Razones por las cuales fallan los proyectos de software:

1. Escasa ***planificación*** del proyecto y ausencia de dirección.

2. ***Comunicación*** insuficiente.

3. ***Administración*** poco efectiva.

4. Fallas en la alineación y ***sinergia*** entre las partes interesadas.

5. Ausencia de apoyo por parte de la ***Dirección*** o Administración Ejecutiva.

6. Equipo con pocas ***habilidades*** técnicas.

7. Ausencia de uso de ***metodologías*** o herramientas

Puntos de función

A mediados de 1970 Allan J. Albrecht, mientras trabajaba para IBM, necesitaba un método de estimación de tamaño de software ***independiente del lenguaje de programación.*** En 1979 publicó un artículo donde presentó el modelo denominado Puntos Función. Durante la década de los ‘80 esta métrica fue adoptada en IBM con buenos resultados.

C:\Users\maris\Downloads\PF.mp4

La metodología de Puntos Función involucra 5 categorías funcionales de un software:

1. ***Entradas***: comandos que aceptará el software
2. ***Salidas***: tipo de información que el sistema puede generar
3. ***Consultas***: diferente tipo de información que el usuario puede solicitar al sistema
4. ***Archivos***: cantidad de archivos que se puede acceder al mismo tiempo
5. ***Interfaces***: cantidad de enlaces con otros sistemas

Cada función es calificada según su complejidad en baja, media o alta, y multiplicada por un factor que corresponde al peso de su complejidad

Como los FP se definen desde ***el punto de vista del cliente***, se tiene en cuenta elementos que pueden ser identificados por un usuario externo. Esta propiedad permite definir una métrica que ***no depende de la tecnología.*** Además, los FP se utilizan para predecir el esfuerzo de una aplicación a ser desarrollada.

Puntos de casos de uso

Con el surgimiento del paradigma ***orientado a objetos*** y el modelado de Casos de Uso, se llevaron a cabo investigaciones sobre la factibilidad de implementar metodologías de estimación basadas en diagrama de clases de objetos y casos de uso. Una de estas investigaciones dio origen al método de ***Puntos de Casos de Uso*** (UCPs) creado por Gustav Karner. El método toma ideas de los Puntos Función (PF) de Albrecht de 1979.

El modelo de Puntos de Caso de Uso definido por Karner, es una metodología para estimar el tamaño del software a construir, el cual parte del modelo de Casos de Uso para obtener un valor denominado Puntos de Casos de Uso, que luego puede ser mapeado en valores horas/hombre. Esta metodología toma las bases del modelo de Puntos Función, propuesto por Albrecht

Clasificación de Actores

Complejidad	Descripción	Peso
Simple	Cuando el actor representa otro sistema el cual está definido por medio de una interfaz API	1
Media	Cuando el actor interactúa con otro sistema a través de un protocolo o si es una persona interactuando con una terminal en línea	2
Alta	Cuando el actor interactúa a través de una interfaz gráfica de usuario (GUI)	3

Clasificación de Casos de Uso

Complejidad	Descripción	Peso
Simple	Si posee 3 o menos transacciones incluyendo cursos alternativos o caso de uso implementado con menos de 5 objetos de análisis	5
Medio	Si posee de 3 a 7 transacciones incluyendo cursos alternativos o caso de uso implementado utilizando entre 5 y 10 objetos de análisis	10
Alta	Si posee más de 7 transacciones incluyendo cursos alternativos o caso de uso implementado con más de 10 objetos de análisis.	15

Factores de Complejidad Técnica

F_i	Factor de Complejidad	Peso
F1	Sistema Distribuido	2
F2	Buen nivel de performance en cuanto al nivel de procesamiento y tiempos de respuesta	1
F3	Eficiencia del usuario final	1
F4	Complejo procesamiento interno	1
F5	Reusabilidad de código	1
F6	Fácil instalación	0.5
F7	Fácil utilización y comprensión del sistema	0.5
F8	Portabilidad	2
F9	El sistema es flexible y abierto a modificaciones	1
F10	Administra altos niveles de concurrencia	1
F11	Características especiales de seguridad	1
F12	Inclusión de componentes de terceros	1
F13	Capacitación al usuario final	1

Factores de Entorno

	Factor de Entorno	Peso
E1	Familiaridad con Objectory / RUP	1.5
E2	Grupo de trabajo part time	-1
E3	Capacidades del analista	0.5
E4	Experiencia en el dominio de la aplicación	0.5
E5	Experiencia en paradigma OO	1
E6	Motivación del grupo de desarrollo	1
E7	Dificultad del lenguaje de programación	-1
E8	Estabilidad de los requerimientos	2

1) Clasificación de actores
2) Clasificación de casos de uso (por transacciones o clases de análisis
3) Obtengo el Tfactor a partir de factores de complejidad técnica
4) Obtengo factores técnicos TCF=0,6+(0,01*Tfactor)
5) Obtengo factores ambientales a partir de Efactor
6) EF=1,4+(-0,03*Efactor)
7) UCP=UUCP*TCF*EF

Ágil Planning Poker

Recordamos que SCRUM tiene 5 eventos, donde uno de ellos es la Reunión de Planificación. Dicha reunión es la que permite determinar que estimación, además de su priorización, tendrán las distintas Historias de Usuario.

De esta menara se conformará la Pila de Producto para luego permitirnos seleccionar que Historias de Usuario se desarrollaran en cada SPRINT (Pila de Sprint).

Existen distintas técnicas de estimación, una de las más conocidas es la técnica de Plannig Poker, donde cada miembro del equipo de desarrollo podrá indicar el tiempo que considera que tomará realizar cada Historia de Usuario. En este sentido cada miembro aplica si experiencia, conocimiento del negocio, conocimiento en tecnología, etc y todos los integrantes del equipo llegan al consenso de cual será la estimación definitiva de la Historia de Usuario.

Planning Poker consiste en utilizar una serie de cartas que contienen números que sea ascendente no todos continuo como puede ser la serie de fibonachi.

La idea es que en la sesión de estimación cada integrante del equipo posee el mismo juego de cartas.

Procedimiento de estimación con Planning Poker:

1) Se selecciona una Historia de Usuario. Se lee su contenido, donde se debaten aspectos técnicos y de negocio en caso de existir dudas.

2) Cada miembro del Equipo de desarrollo toma una carta de su baraja con el valor que estima que debe tenerse en cuenta para la realización de la Historia de Usuario y la muestra a los miembros.

3) Se comparan los distintos valores para determinar si existe consenso, en dicho caso ese será el valor estimado. Si los valores fueran distintos se analizan los extremos donde los miembros del equipo comentan de su decisión argumentando aspectos técnicos, experiencia, etc. Se repite el proceso desde el punto 2 hasta llegar al conceso necesario del equipo.

Baraja Planning Poker:

La baraja que se utiliza para Planning Poker además de las cartas con la serie de valores que permite estimar suele tener cartas que ayudan a los miembro del equipo a expresar distintas situaciones. Ej.: pausas, historias de usuario muy amplias, dudas sobre la historia de usuario, etc.

BACK LOG PRIORIZADO

	Épicas	Historias de Usuario	ESTIMACIÓN Diego	Steffy	Leno	Juan	Resultado Final
E1	El sistema debe permitir Gestionar datos de Usuario	H_1: Como administrador quiero poder agregar Conductores y Supervisores para Tener un registro del personal que labora	13	20	100 x	5	13
		H_2: Como administrador quiero poder actualizar los datos del Usuario del sistema deshabilitar / habilitar a solicitud del jefe de tramo.	40	40	13	8	40
		H_3: Como administrador quiero poder visualizar todo el personal por datos de Usuario para visualizar los Activos e Inactivos	5	8	20	8	8
E2	El sistema debe permitir registrar los movimientos de los Vehículos Ferroviarios	H_4: Como conducir quiero agregar maniobras que se ejecutan en el Tren para que el supervisor de turno pueda Tener un control de los movimientos	40	40	20	20	40

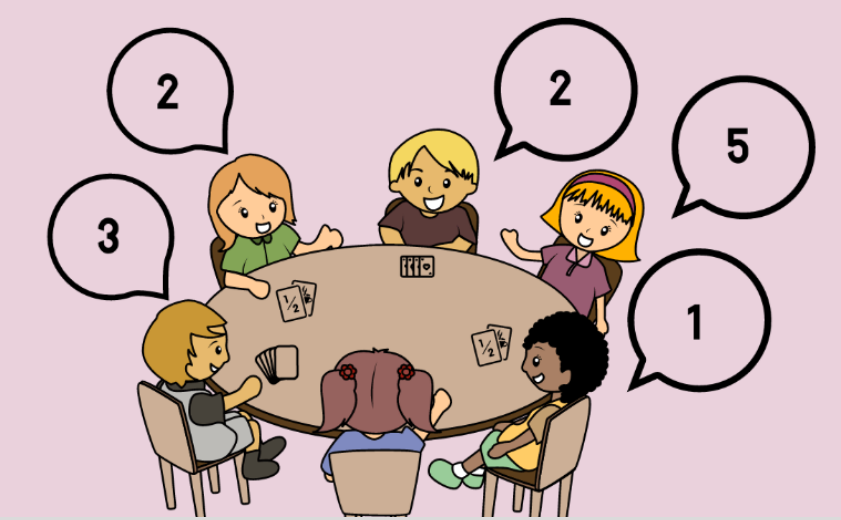
Material de consulta gratuito:

1) Curso de Certificación Academia Kleer.

https://academia.kleer.la/p/estimaciones-agiles

2) Videos Youtube "**Ágil Es - Por Cris Rúa**"

https://www.youtube.com/watch?v=ey6Pm46WXkY



Ágil

Puntos de Historias

Los Puntos de Historias es un técnica de estimación en la cual se utiliza la estimación de una Historia de Usuario como referencia para estimar las restantes Historias del proyecto.

La técnica trata de realizar una aplicación natural como podría suceder en la vida cotidiana, ya que se podría establecer que “algo” es más o menos que un determinado parámetro. Ej.: Se podría decir que el viaje en colectivo es económico con una gaseosa si por ejemplo no me parece costosa la gaseosa y el valor del colectivo es menor. En este caso la unidad de medida es el costo de la gaseosa.

Esto mismo pero en el ámbito de la estimación trata de reflejar está técnica al buscar una Historia de Usuario como referencia. Por lo general se tiene como referencia una Historia de Usuario que se repite a lo largo de los proyecto, ejemplo puede ser el ingreso con usuario y clave a los sistemas informáticos (Login), de esta forma cada historia se podría comparar y determinar por ejemplo cuantas Historias de Usuario de Login representan la historia analizada.

Ejemplo de estimación relativa

Punto de pivote:
Huevo cocinado

Tres minutos

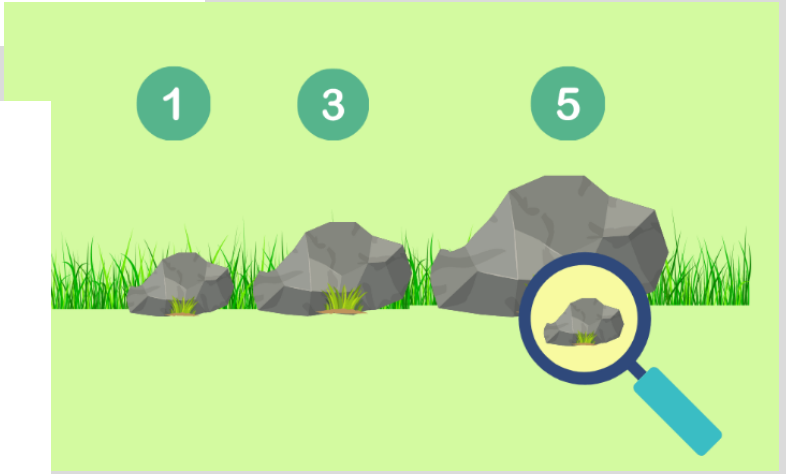
¿En cuántos pivote puedo barrer mi casa?

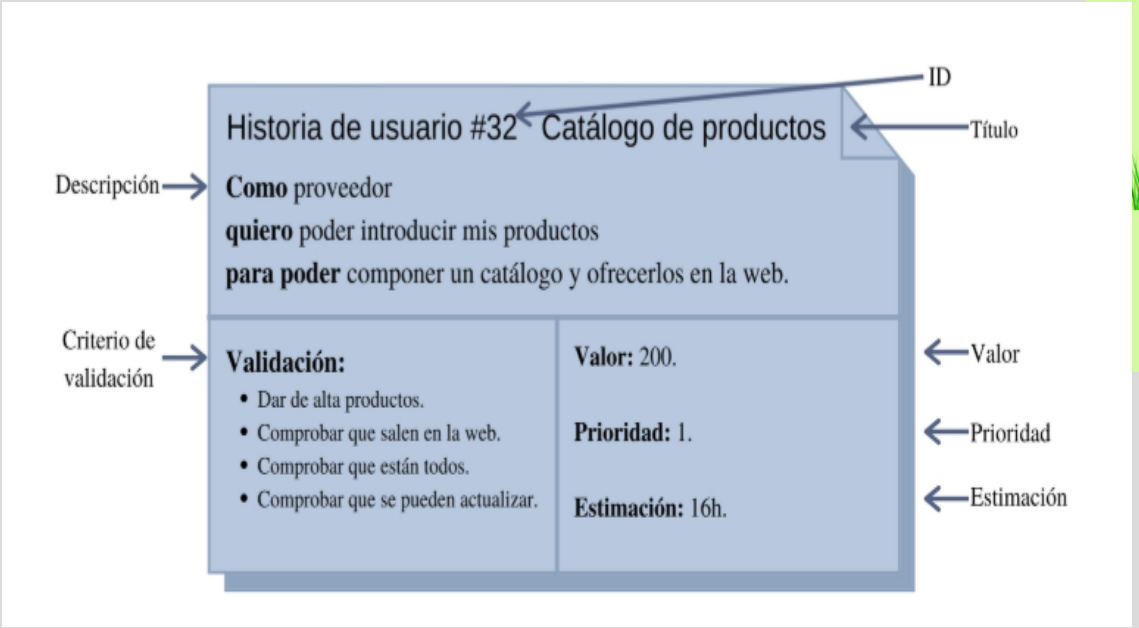
¿En cuántos huevos fritos puedo barrer mi casa?

Si te tardas 15 minutos, quiere decir que te toma 5 puntos de pivote.

=

5 huevos estarán listos cuando termines de barrer.





Material de consulta gratuito:

1) Videos Youtube "**Ágil Es - Por Cris Rúa**"

https://www.youtube.com/watch?v=O-D22kLYi2M

https://www.youtube.com/watch?app=desktop&v=4I6F0nSCdmE