



Estimaciones Ágiles

▼ Diferencia entre Método Tradicional y Ágil

En el desarrollo de software siempre se marcan diferencias entre el **método tradicional** y el **método ágil**, sobre todo cuando hablamos de estimaciones.

👉 **Estimar** significa intentar predecir el futuro, tratar de calcular lo más certeramente posible cuánto tiempo, esfuerzo y recursos se necesitarán para cumplir con una tarea. No es una verdad absoluta, sino una referencia que servirá como base para la planificación.

La **estimación** es una técnica o conjunto de técnicas para planificar, pero no es lo mismo que la **planificación** en sí. Estimar me ayuda a tener una idea de cuánto llevará implementar una user story, pero no debería comprometerme como si fuera palabra santa.

De hecho, la estimación actúa como una **protección para el equipo**: si una tarea me lleva al menos 150 horas lineales, en el calendario la planificación no puede reflejar menos de eso. Siempre debe ser mayor o igual.


▼ Estimaciones Tradicionales

En los enfoques tradicionales, las estimaciones se hacen sobre el **producto a construir**. Se parte de calcular:

1. **El tamaño del software** → por ejemplo, cuántos casos de uso tendrá.
2. **El esfuerzo** → medido en horas lineales (hora-hombre).
3. **El tiempo de calendario** → considerando superposición de tareas, cantidad de personas y horas disponibles de cada una.

De esta manera, se pasa de estimaciones absolutas (ej: 15 horas lineales) a estimaciones en calendario (ej: 2 días considerando el equipo disponible).

 Unidad de esfuerzo: **hora lineal**.

 Unidad de calendario: **día**.

También se pueden usar técnicas de **analogía**, comparando con proyectos similares ya realizados.

▼ Estimaciones Ágiles ⚡

En Agile las cosas cambian. No se hacen estimaciones absolutas, sino **relativas**. Esto significa que el valor solo tiene sentido dentro del equipo que lo definió. Para esto se utiliza una unidad de medida especial: los **Story Points (SP)**.

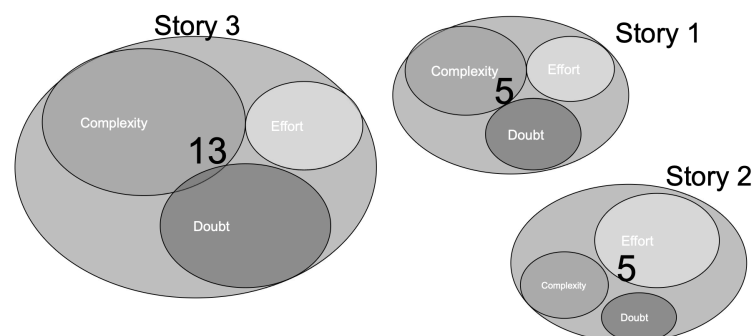
Los Story Points consideran tres variables:

1. **Complejidad** de la user story o feature.
2. **Trabajo** que demandará.
3. **Incertidumbre** que genera.

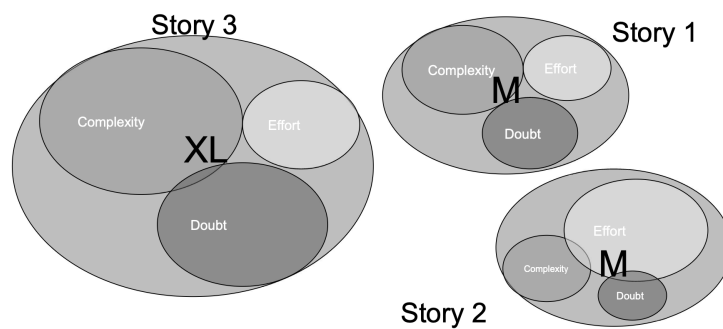
La combinación de estas variables se pondera de manera subjetiva, por consenso del equipo. Es importante destacar que en este contexto, **tamaño ≠ esfuerzo**.

Para darle escala, se utiliza la **serie de Fibonacci** (1, 2, 3, 5, 8, 13...). La razón es que el esfuerzo y la complejidad crecen de forma exponencial a medida que aumenta la dificultad de las user stories.

Y si usamos números?



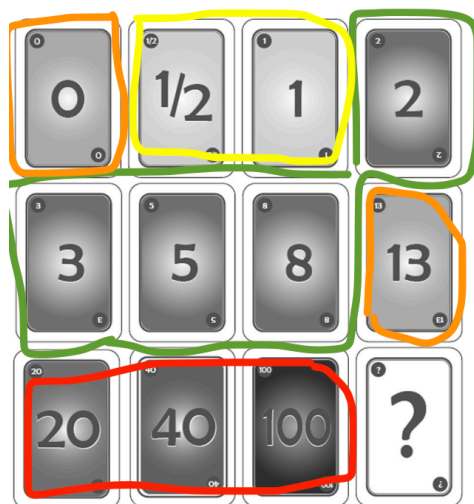
“tamaño” de las Stories



El equipo siempre parte de una **user story base o canónica** (la más simple) que se define como de 1 SP. A partir de ahí, comparan las demás user stories preguntándose: “¿cuántas veces más grande es esta respecto de la base?”.

👉 Si una user story recibe un **0** o un **“?”**, no es estimable y no cumple con el criterio INVEST de estar lista.

👉 Si recibe un **13**, debe partitionarse porque no cumple con el criterio de **Small**.



- **0**: Quizás ud. no tenga idea de su producto o funcionalidad en este punto.
- **1/2, 1**: funcionalidad pequeña (usualmente cosmética).
- **2-3**: funcionalidad pequeña a mediana. Es lo que queremos. 😊
- **5**: Funcionalidad media. Es lo que queremos 😊
- **8**: Funcionalidad grande, de todas formas lo podemos hacer, pero hay que preguntarse sino se puede partir o dividir en algo más pequeño. No es lo mejor, pero todavía 😊
- **13**: Alguien puede explicar por que no lo podemos dividir?
- **20**:Cuál es la razón de negocio que justifica semejante story y más fuerte aún, por qué no se puede dividir?.
- **40**: no hay forma de hacer esto en un sprint.
- **100**: confirmación de que está algo muy mal. Mejor ni arrancar.

▼ Poker Estimation

Una de las técnicas más usadas para asignar Story Points es el **Planning Poker**. Aquí, cada desarrollador propone un valor, se muestran las cartas y se debate especialmente por qué alguien eligió el valor más alto y otro el más bajo. Se realizan rondas hasta que el número converge.

La clave está en la **conversación**: el Product Owner participa para escuchar, pero no puede opinar en la estimación porque no conoce la complejidad técnica del código. El consenso debe darse dentro del equipo de desarrollo (generalmente de 5 ± 2 personas).

Las estimaciones se realizan durante la **ceremonia de Planning**. Solo se discuten las user stories priorizadas en el Product Backlog para el próximo sprint, aplicando el concepto de *just in time*.

▼ Velocidad del Equipo 🚀

Una vez que termina la iteración, se cuentan las user stories aprobadas por el Product Owner y se suman sus Story Points. Ese total define la **velocidad del equipo**.

Por ejemplo: si se aprobaron user stories equivalentes a **16 SP**, esa es la velocidad. Al inicio la velocidad es inestable, pero después de 4 o 5 sprints suele estabilizarse.

Con esta medida, el equipo puede planificar mejor los siguientes sprints, sabiendo cuántas user stories incluir de acuerdo a su capacidad (ej: 16 ± 2 SP).

◆ Importante: en el **Sprint Backlog** se almacenan solo las user stories aprobadas por el Product Owner y las tareas asociadas. En el **Product Backlog**, en cambio, no se detallan tareas, sino únicamente los ítems de producto.

📖 Recomendación: leer el **Manual de Sprint** para profundizar en estos conceptos.