

# Calidad en contextos ágiles

Darío Macchi



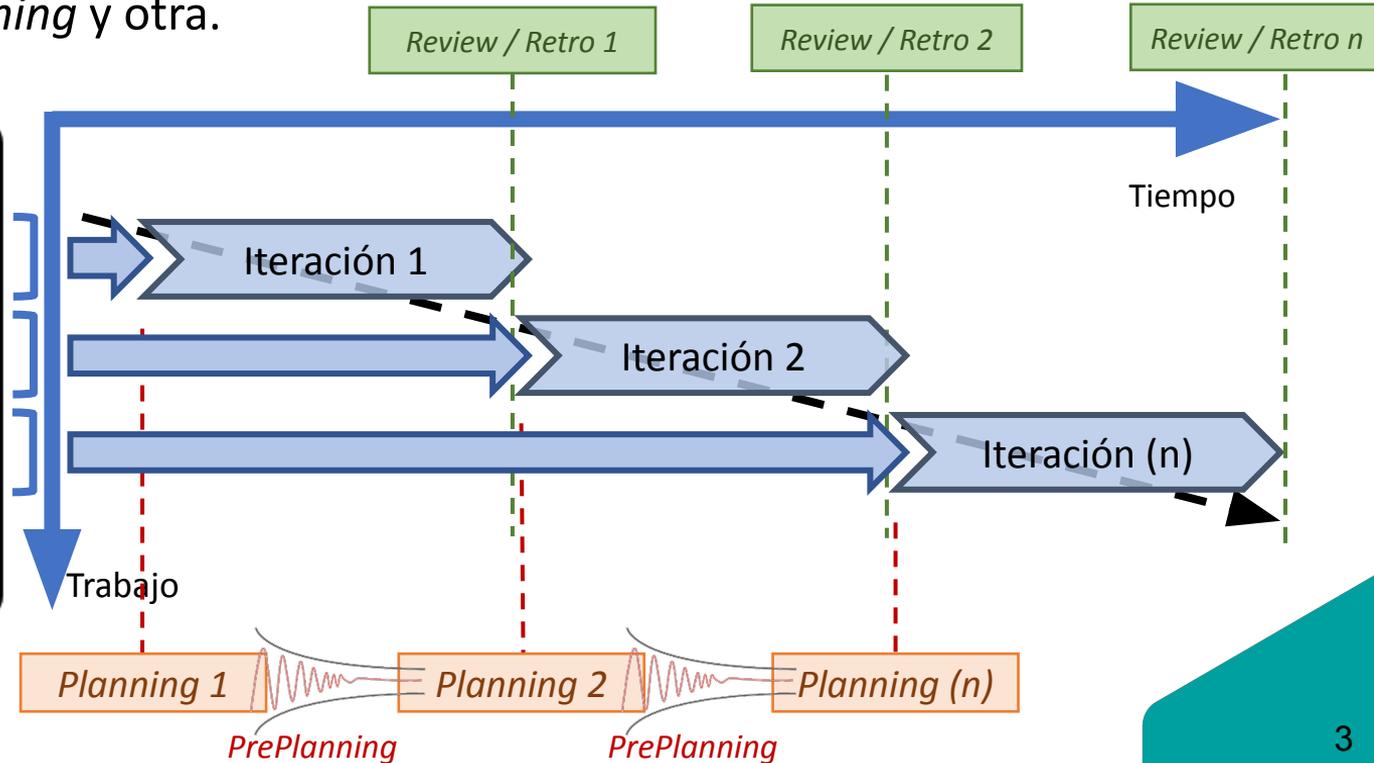


vs.



# Método iterativo (ágil?)

- Iteración: ciclo simple de desarrollo que tiene por duración el tiempo entre una sesión de *planning* y otra.



# Just in Case (JIC) vs. Just in Time (JIT)



**Purchase all we can,  
just in case.**

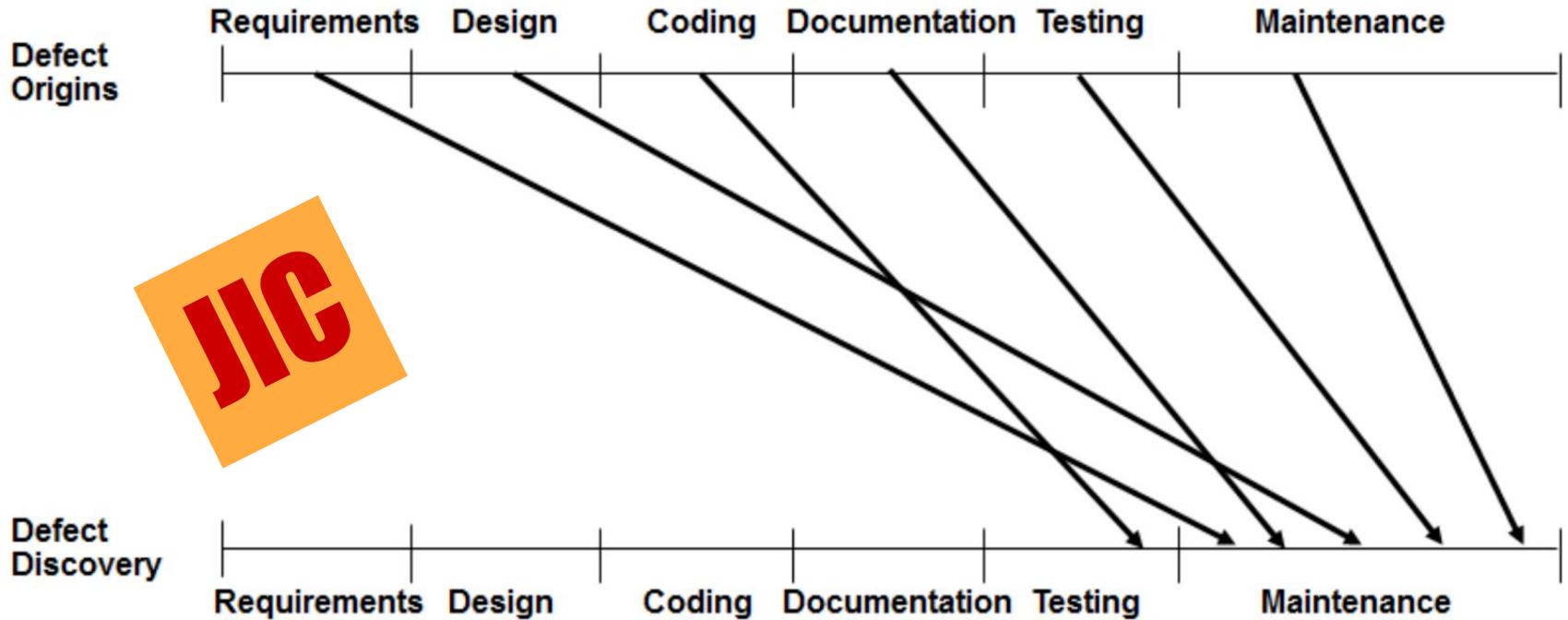
- Maximizing inventory
- Larger inventory orders
- Managing unpredictable demand
- Mitigating supply chain disruptions



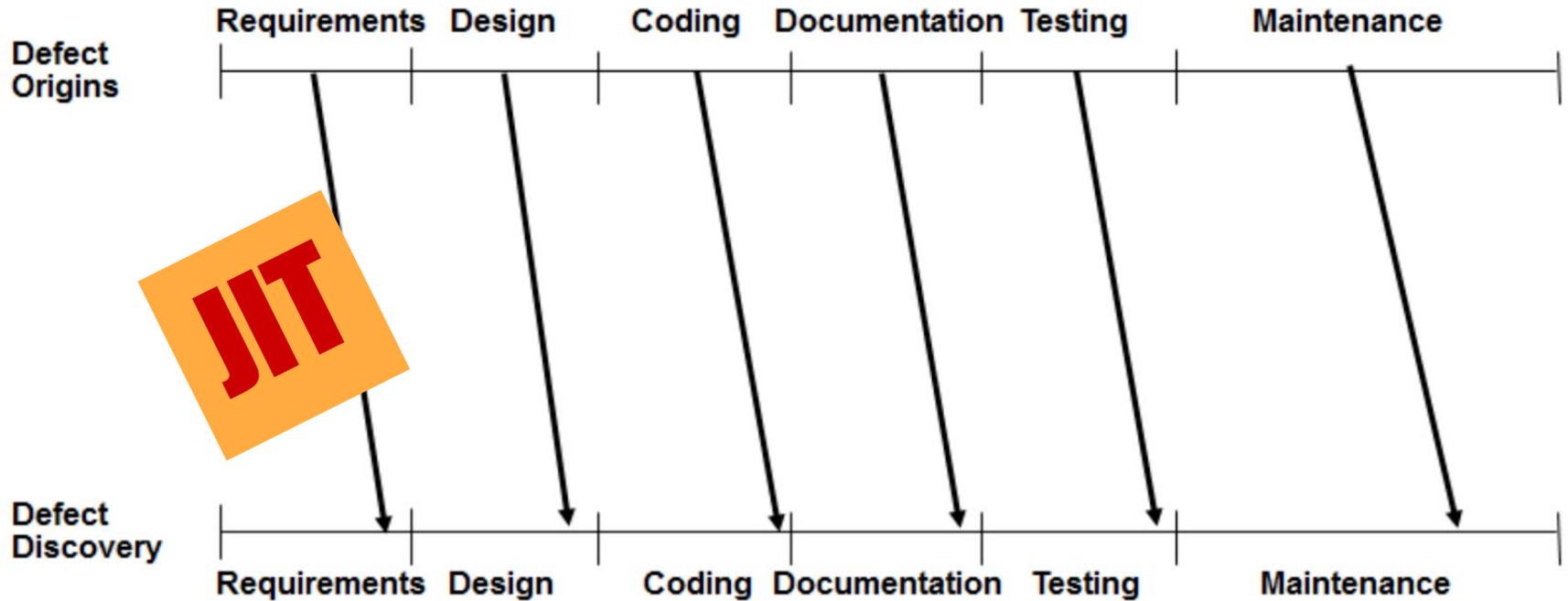
**Only purchase what we need  
when we need it.**

- Minimizing inventory
- Smaller inventory orders
- Accurately forecasting demand
- Sensitive to supply chain disruptions

# Origen/descubrimiento de defectos



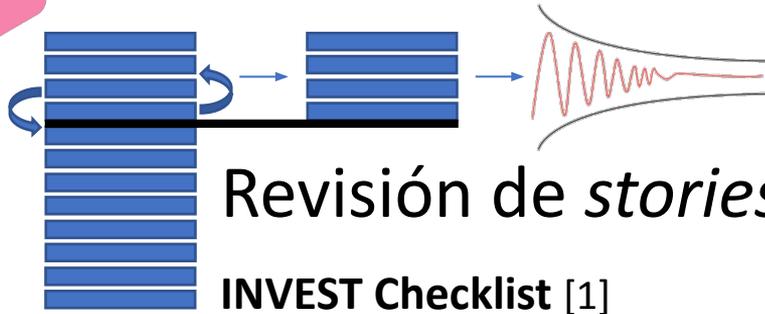
# Origen/descubrimiento de defectos (↗calidad)



# Preplanning



# Preplanning - *Definition of Ready*



- Independent*
- Negotiable*
- Valuable*
- Estimable*
- Small*
- Testeable*

# Preplanning - *Definition of Ready*



Revisión de *stories* del backlog:

INVEST Checklist [1]

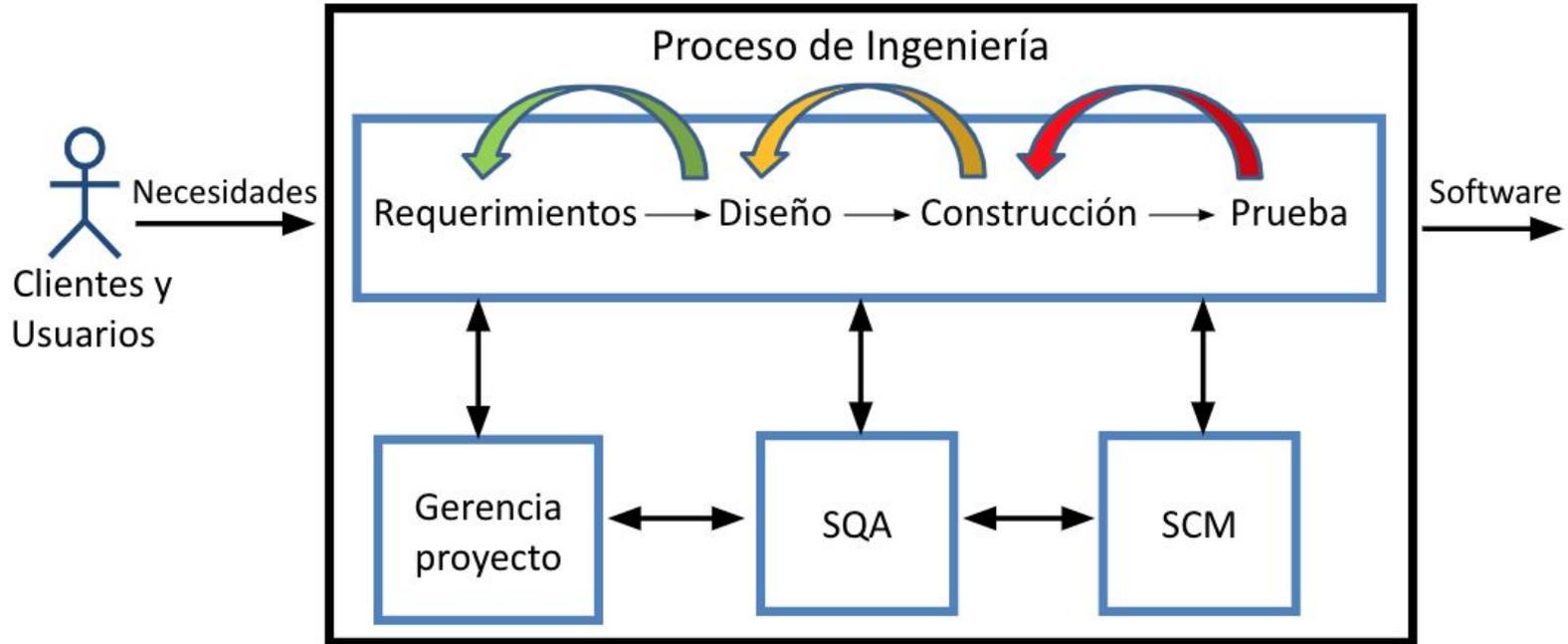
- Independiente
- Pequeño

Testable

**Ahorrar tiempo de planning !!!**

[1] Vogel, B.: INVEST in Good Stories, and SMART Tasks, 2003

# Preplanning - *Shift Left*



# Preplanning - *Shift Left*

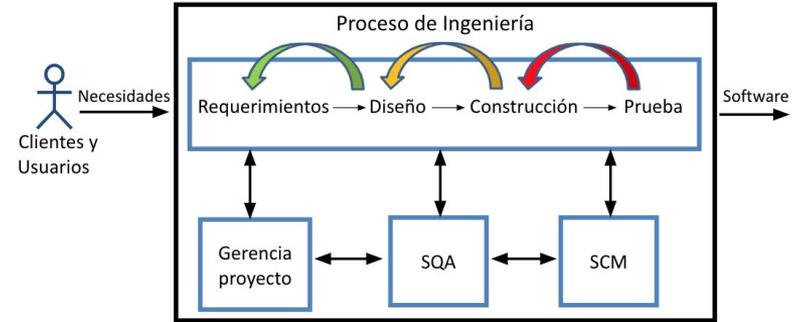
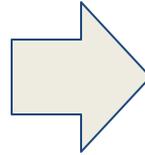
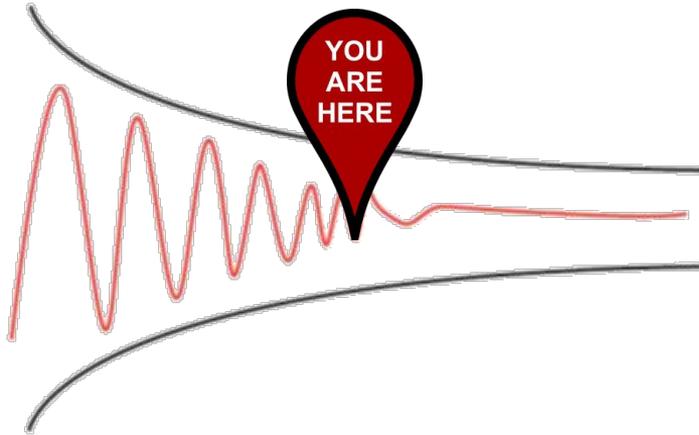
Involucrar *testers* en etapas tempranas para:

- Aprender: *testing* es evaluar un producto **aprendiendo** de él mediante la exploración y la experimentación.
- Desafiar otros roles: agregar escepticismo, anticipar problemas (*Doomsayer* de XP?) y encontrar puntos donde nos estemos “engañando” utilizando pensamiento crítico.
- Perseguir la *testeabilidad*
- *Pair testing*: desarrollador explica el código al tester. Luego este puede hacer preguntas y finalmente probar la funcionalidad.

# Planning



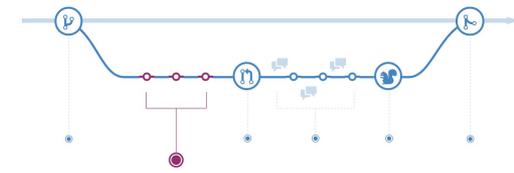
# Planning



# Coding



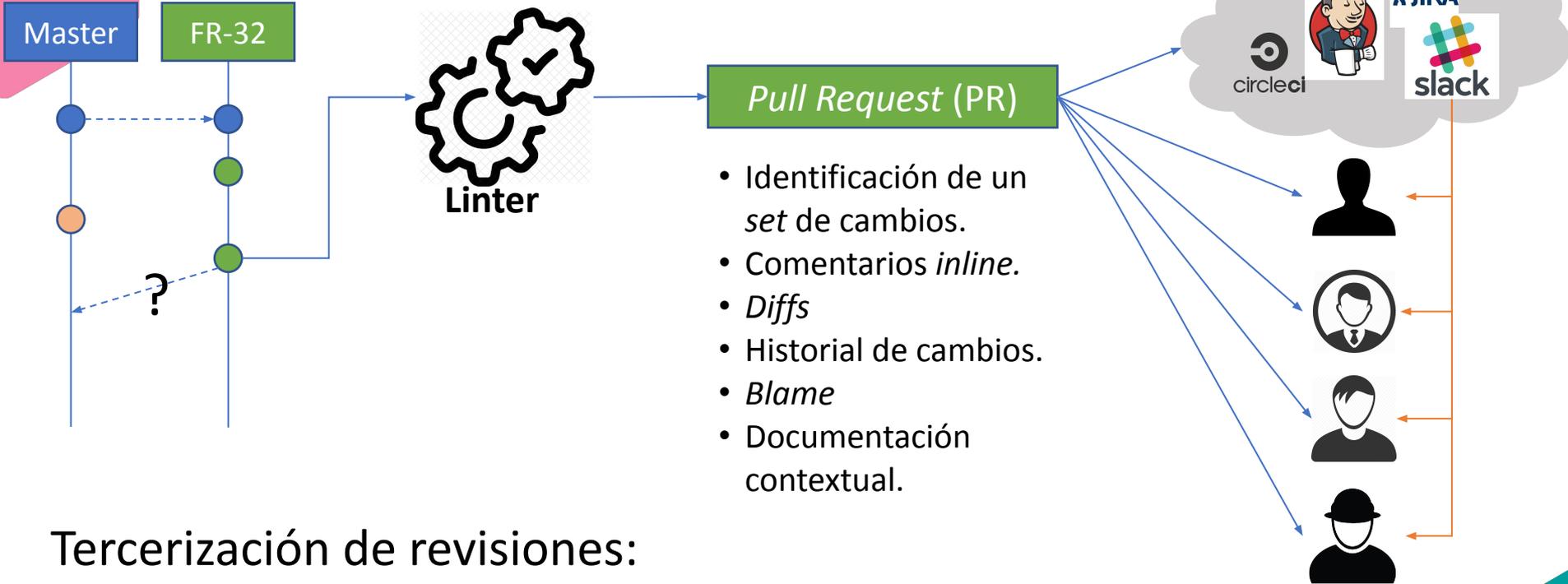
# Coding - Feature



ANTES de un PR... revisión propia del artefacto de código terminado



# Coding - Feature



## Tercerización de revisiones:

- <https://www.reddit.com/r/codereview/> - Canal de solicitudes de *reviews*
- <https://www.pullrequest.com/> - Code Review as a Service

# Coding - Feature

```
Show more
770 798     public ApplicationUser getUser() {
771 799         return user;
772 800     }
773 801
774 802     public Optional<Long> getExistingProjectId() {
775 803         return existingProjectId;
776 804     }
777 805 }
778 806
779 807 @PublicApi
780 - public static class UpdateProjectValidationResult extends AbstractProjectValidationResult {
808 + class UpdateProjectValidationResult extends AbstractProjectValidationResult {

Bob Foo
Could you explain why there doesn't need to be a public static here?
Reply · Create task · Like · 18 Feb 2016

Alice Bar
A class defined in an interface is public/static by default/definition.
Reply · Create task · Like · 19 Feb 2016

781 809     private final Project originalProject;
810 + private final UpdateProjectRequest updateProjectRequest;
782 811
783 812     @Internal
784 813     public UpdateProjectValidationResult(ErrorCollection errorCollection) {
785 814         super(errorCollection);
786 815         this.originalProject = null;
816 +         this.updateProjectRequest = null;
787 817     }
788 818
789 819     @Deprecated
```

# Coding - Feature



```
127 + private boolean canPlayVideo(final Mode mode) {  
128 +     return !(playerMediator.isVideoPlayingOrLoading() || !settings.isA
```



**dpreussler** on Mar 6



is there a test for that? You need to be a vulcanian to read that?



## Coding - Feature

**If In JIRA it is not,**

**exist it does not!**

# Coding - Feature

## Antes

- Tickets de calidad para saber qué hacer y porqué

## Durante

- ~~Documentar~~ Dejar notas para el yo del futuro
  - Qué estaba pensando?
  - Qué opciones había y por qué elegí lo que elegí
  - Qué información de contexto tenía?
  - Qué restricciones?

## Después

- Herramienta de consulta
- Histórico de decisiones
- Transferencia de conocimiento





[austinsoftware.com](http://austinsoftware.com)

