Maestría en Sistemas Embebidos

Universidad de Buenos Aires

Sistema de control para estación autónoma marítima de monitoreo de ruido ambiente

Esp. Ing. Patricio Bos

Director:
Dr. Ing Ariel Lutenberg

Jurados:
Dr. Ing. Pablo Gómez
Ing. Juan Manuel Cruz
Mg. Lic Igor Prario

Agenda

Motivación

Planificación

Metodología

Implementación

Testing

Demo

Conclusiones

Agenda

- Motivación
- Planificación
- Metodología
- Implementación
- Testing
- Demo
- Conclusiones

Motivación

- ¿Por qué acústica submarina?

Motivación

- ¿Por qué acústica submarina?
- ¿Qué es el nivel de ruido?

Motivación

- ¿Por qué acústica submarina?
- ¿Qué es el nivel de ruido?
- ¿Por qué interesa medirlo?

Motivación

- ¿Por qué acústica submarina?
- ¿Qué es el nivel de ruido?
- ¿Por qué interesa medirlo?
- ¿Qué disciplinas lo necesitan?

Antecendentes

Objetivo

- ¿Por qué acústica submarina?

Objetivo

- ¿Por qué acústica submarina?
- ¿Qué es el nivel de ruido?

Objetivo

- ¿Por qué acústica submarina?
- ¿Qué es el nivel de ruido?
- ¿Por qué interesa medirlo?

Objetivo

- ¿Por qué acústica submarina?
- ¿Qué es el nivel de ruido?
- ¿Por qué interesa medirlo?
- ¿Qué disciplinas lo necesitan?

Agenda

Motivación

Planificación

Metodología

Implementación

Testing

Demo

Conclusiones

Diagrama en bloques

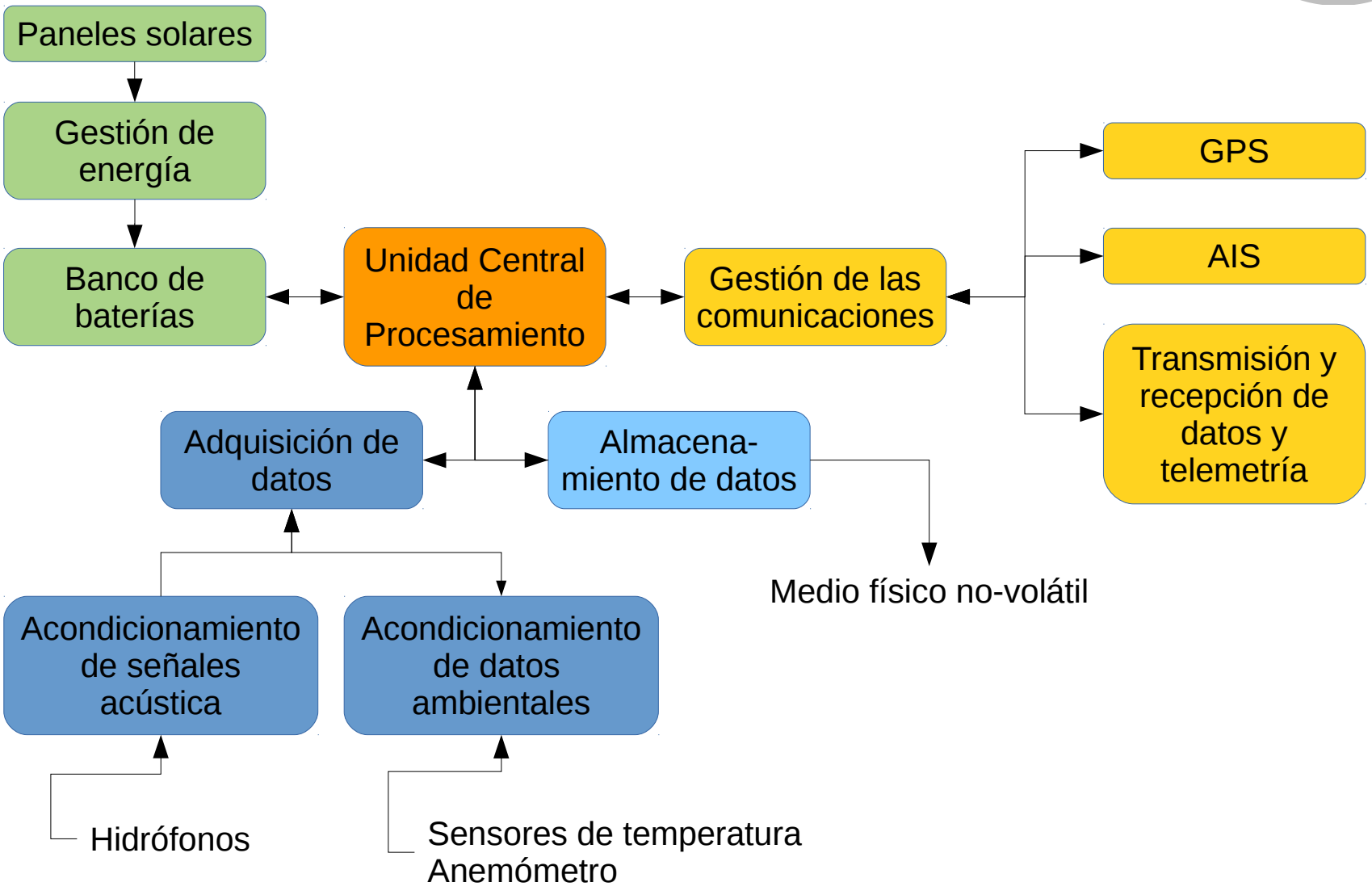
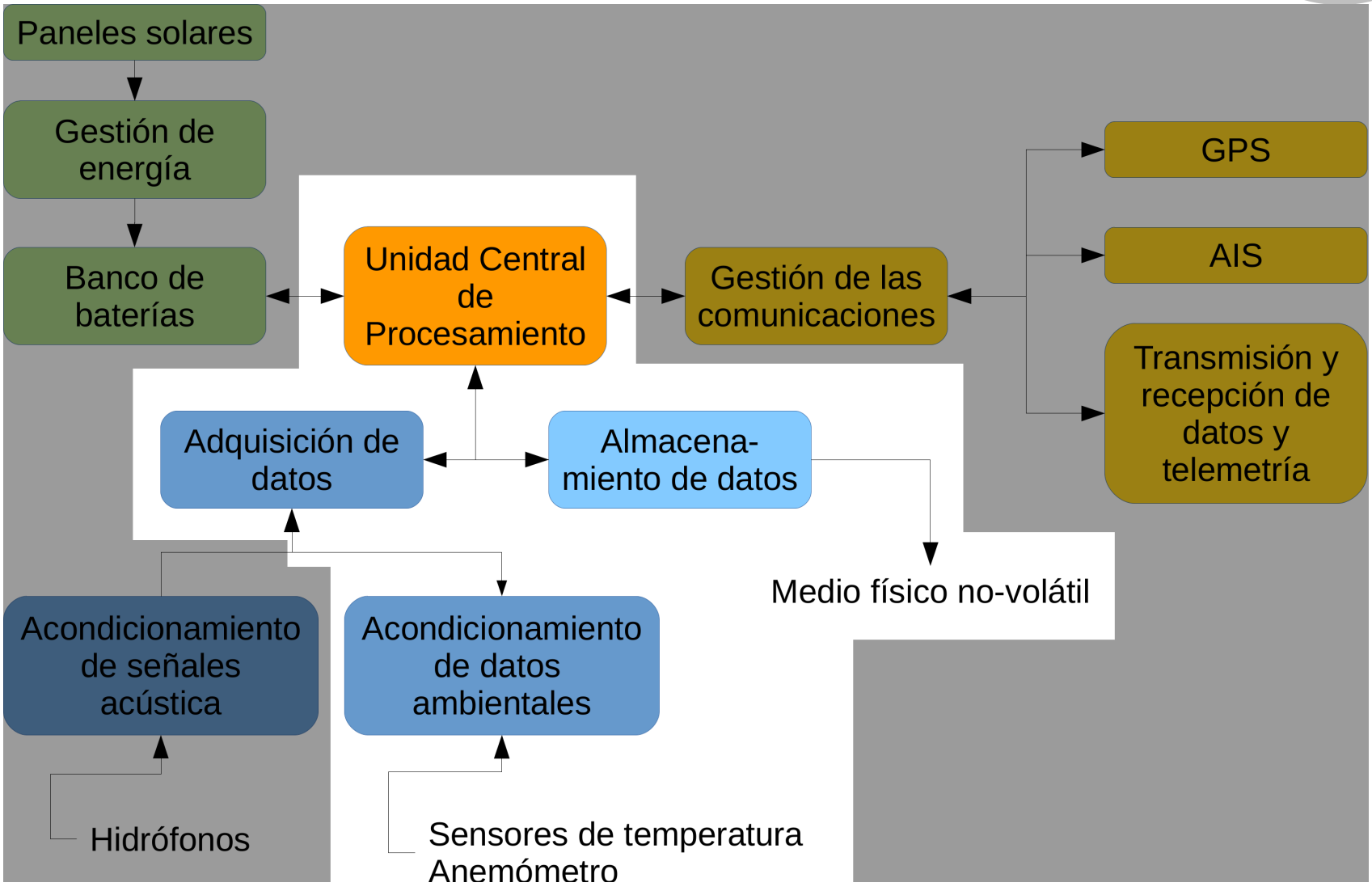


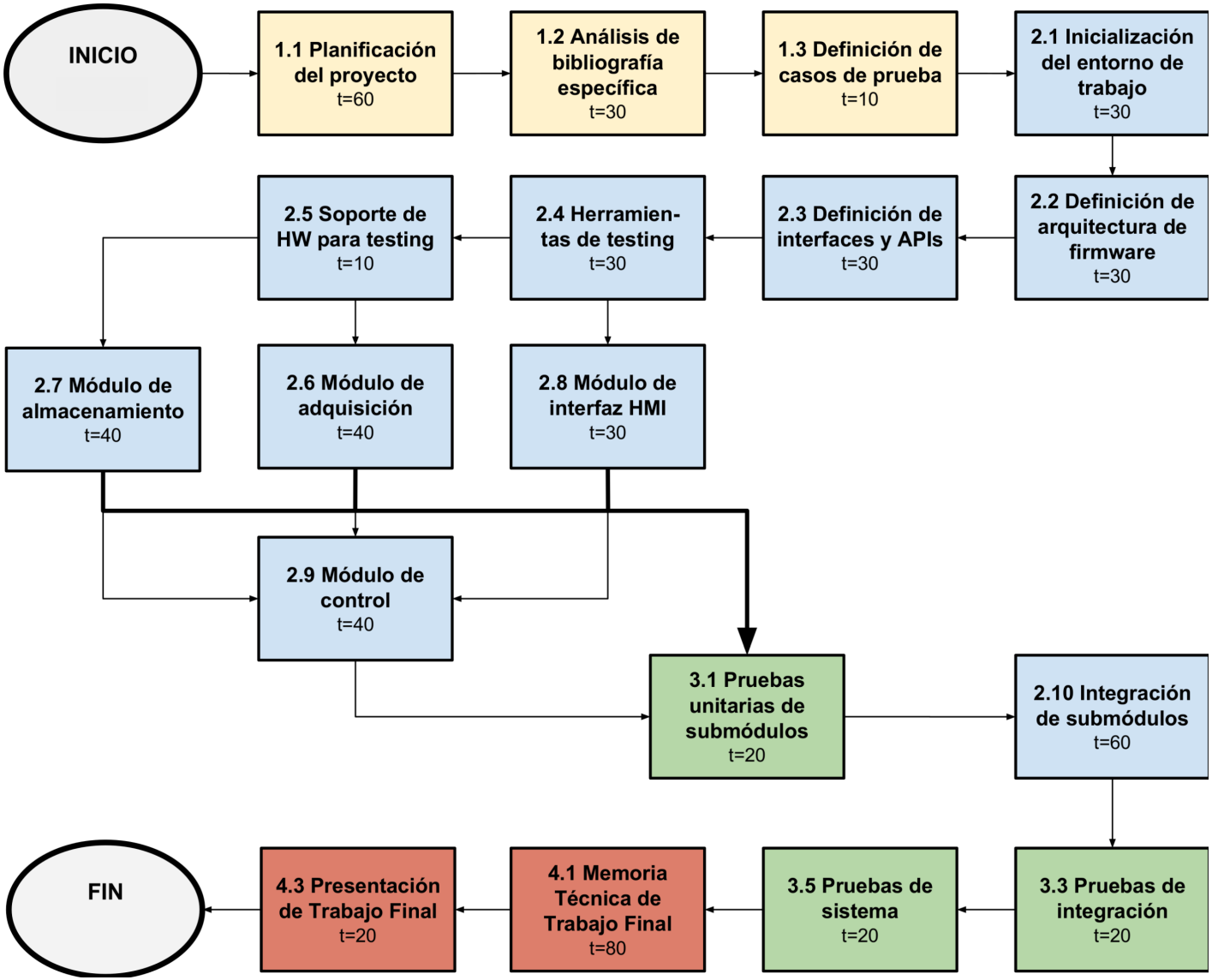
Diagrama en bloques



Planificación en etapas

Etapa	Horas	Hitos
Documentación y análisis preliminar	100	Plan de trabajo
		Presentación de plan de trabajo
Diseño e implementación	340	Documentación de submódulos
Verificación y validación	60	Reporte de pruebas unitarias
		Reporte de pruebas de integración
		Reporte de resultados de casos de prueba
Proceso de cierre	100	Memoria Técnica
		Presentación de Trabajo Final

Desglose de tareas en AoN



Agenda

Motivación

Planificación

Metodología

Implementación

Testing

Demo

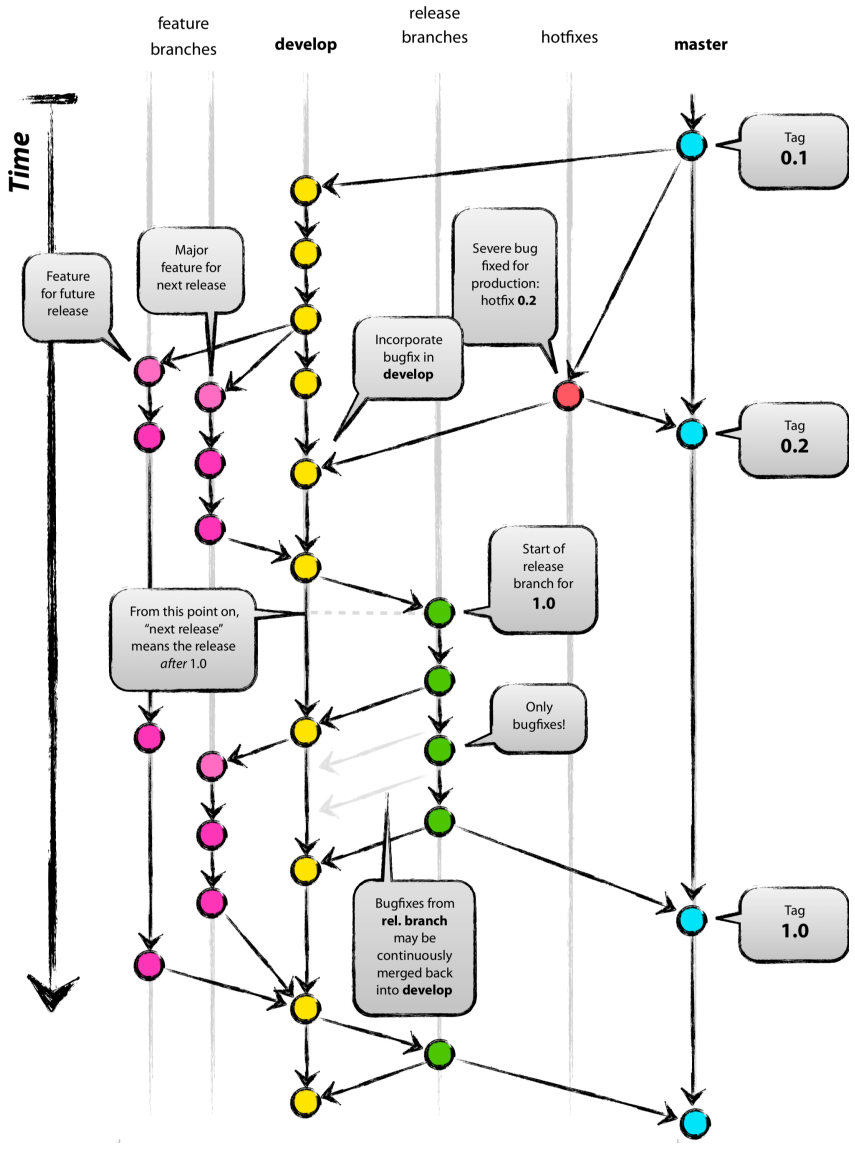
Conclusiones

Modelo de ramas

Successfull git branch model

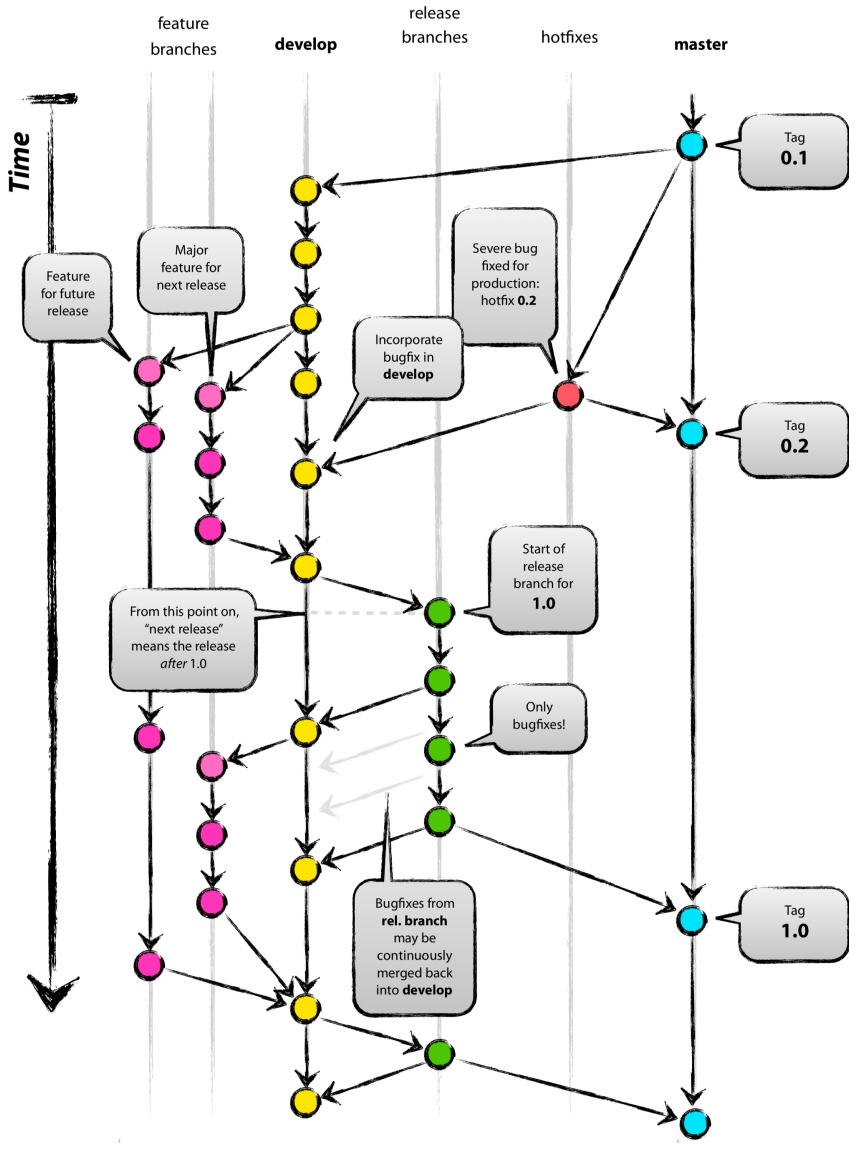
Modelo de ramas

Successfull git branch model



Modelo de ramas

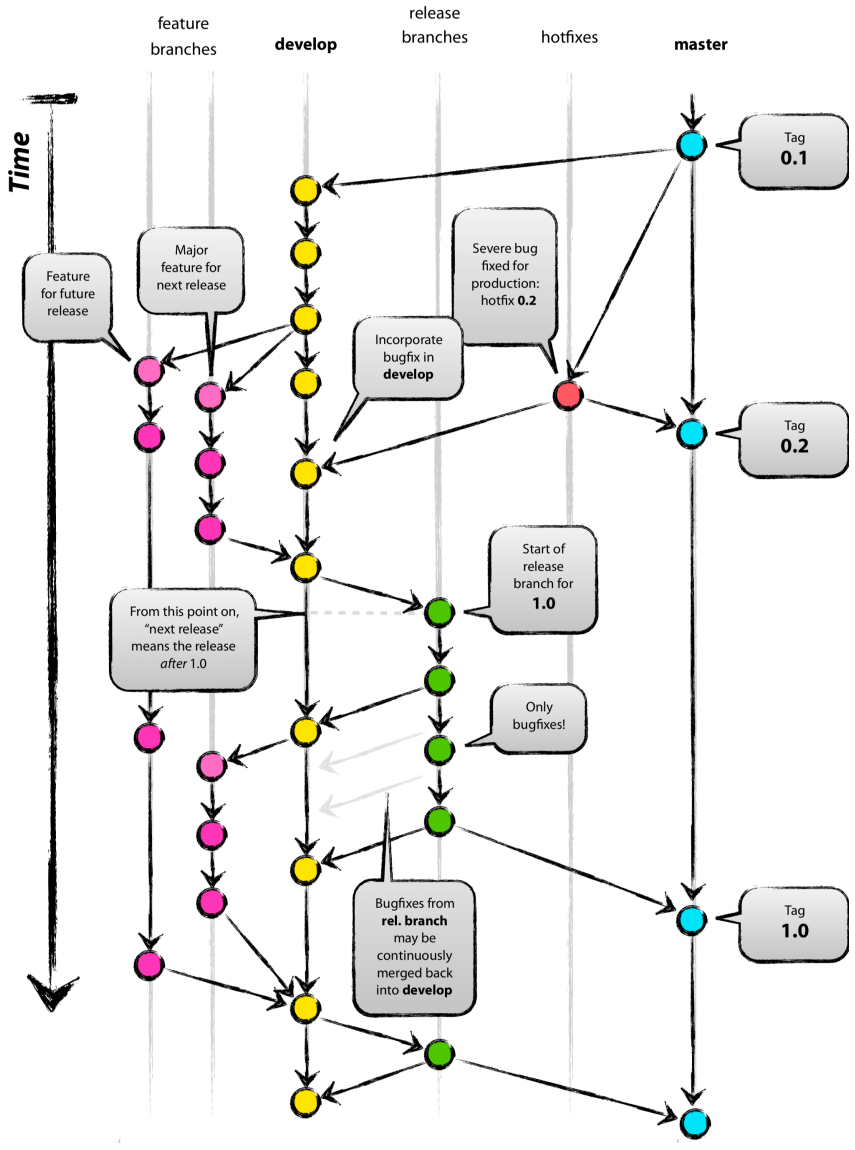
Successfull git branch model



Ramas creadas<+(1)->

Modelo de ramas

Successfull git branch model

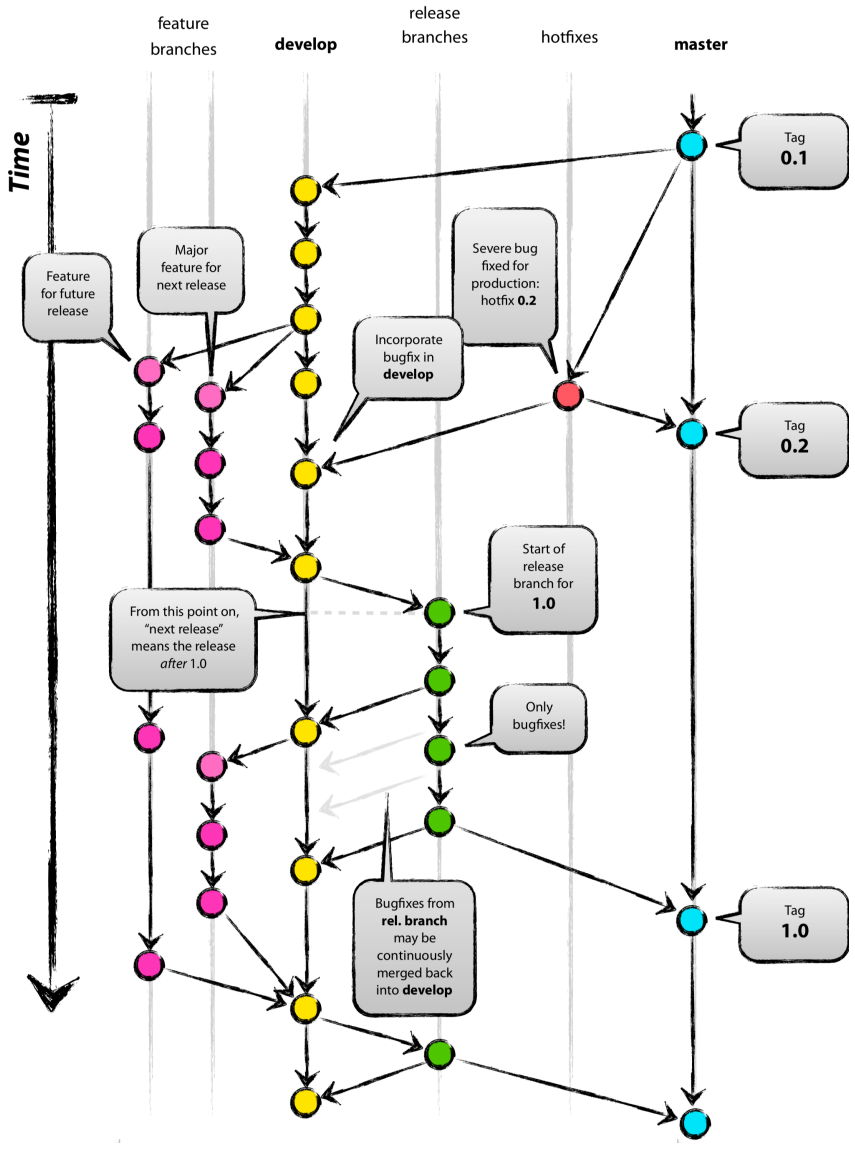


Ramas creadas<+(1)->

- Master

Modelo de ramas

Successfull git branch model

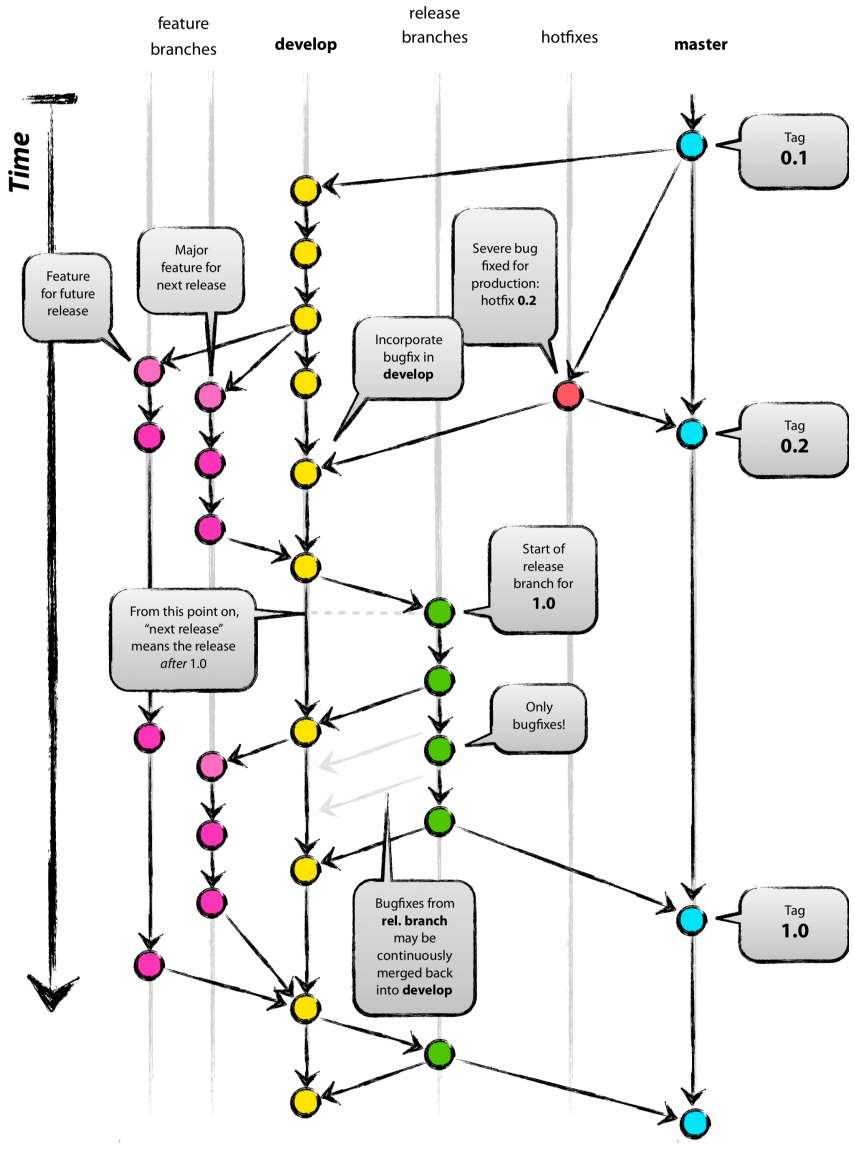


Ramas creadas<+(1)->

- Master
- Develop

Modelo de ramas

Successfull git branch model

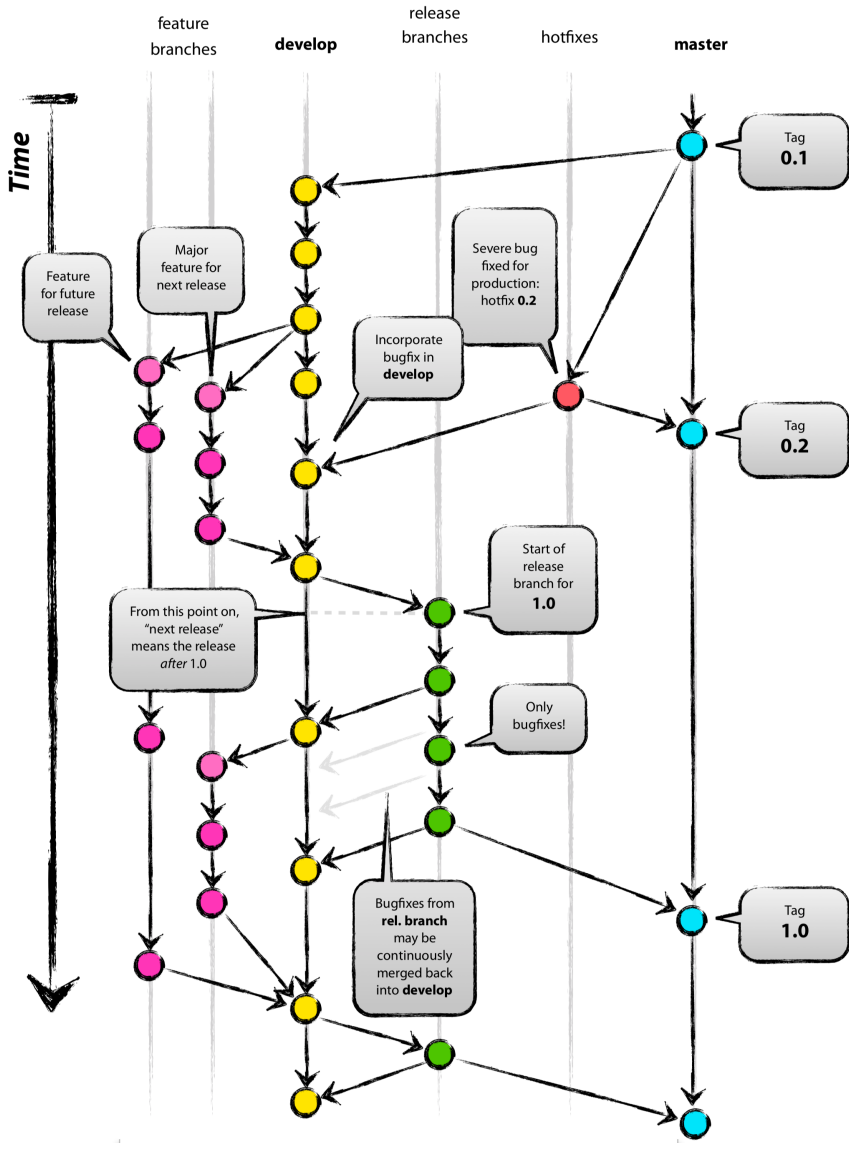


Ramas creadas<+(1)->

- Master
- Develop
- Adquisición

Modelo de ramas

Successfull git branch model

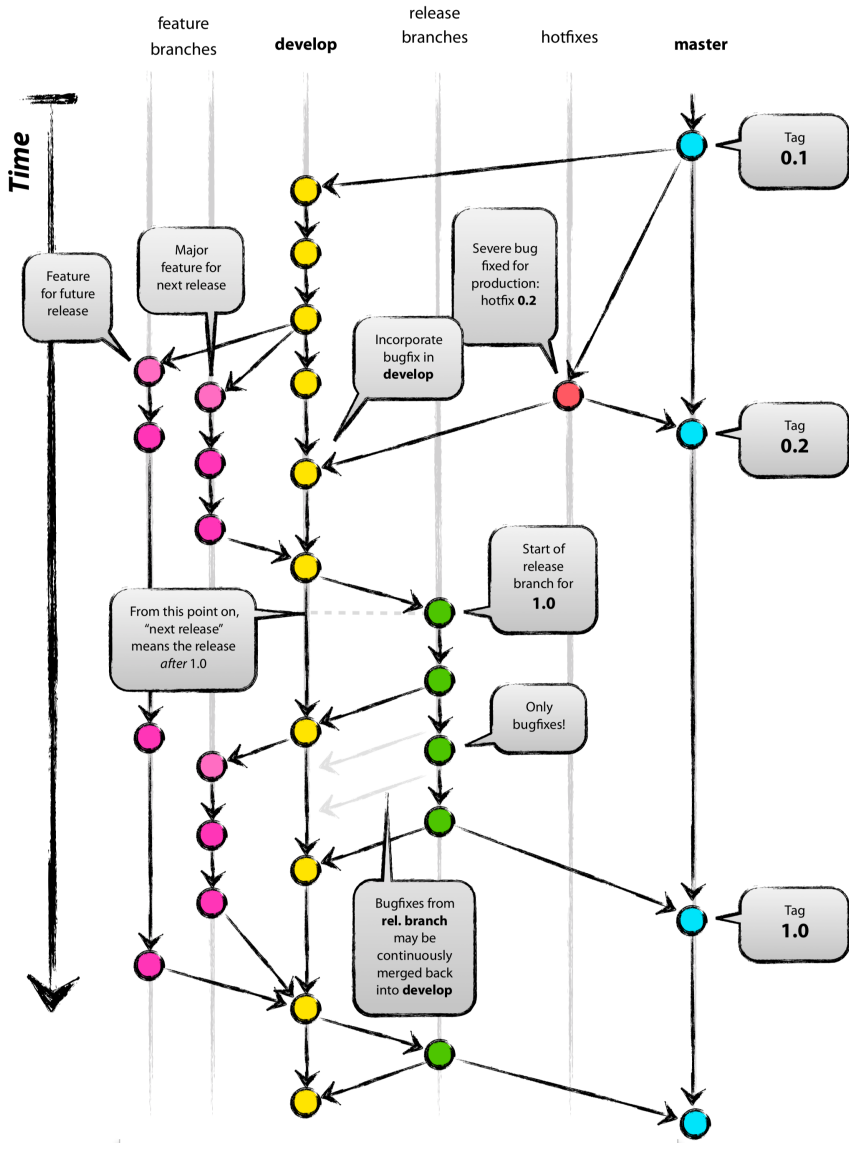


Ramas creadas<+(1)->

- Master
- Develop
- Adquisición
- Almacenamiento

Modelo de ramas

Successfull git branch model

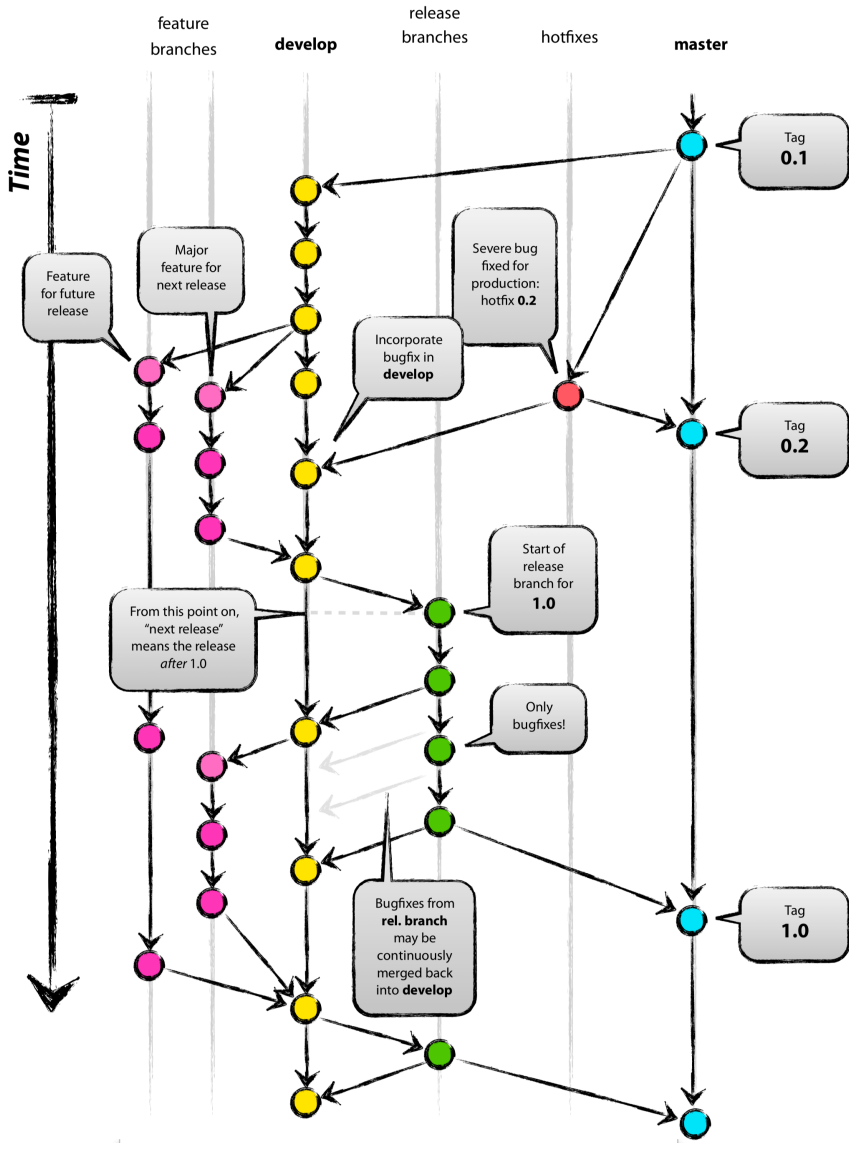


Ramas creadas<+(1)->

- Master
- Develop
- Adquisición
- Almacenamiento
- Interfaz de usuario

Modelo de ramas

Successfull git branch model

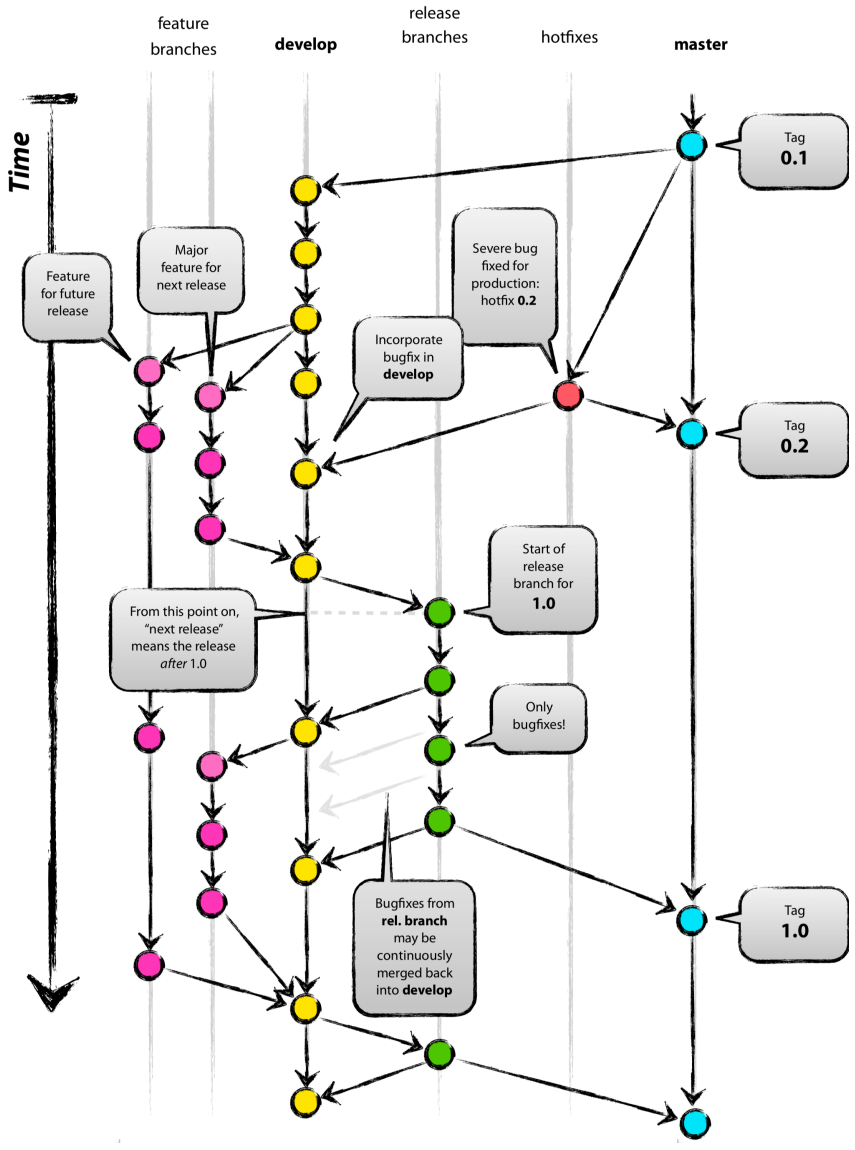


Ramas creadas<+(1)->

- Master
- Develop
- Adquisición
- Almacenamiento
- Interfaz de usuario
- Control

Modelo de ramas

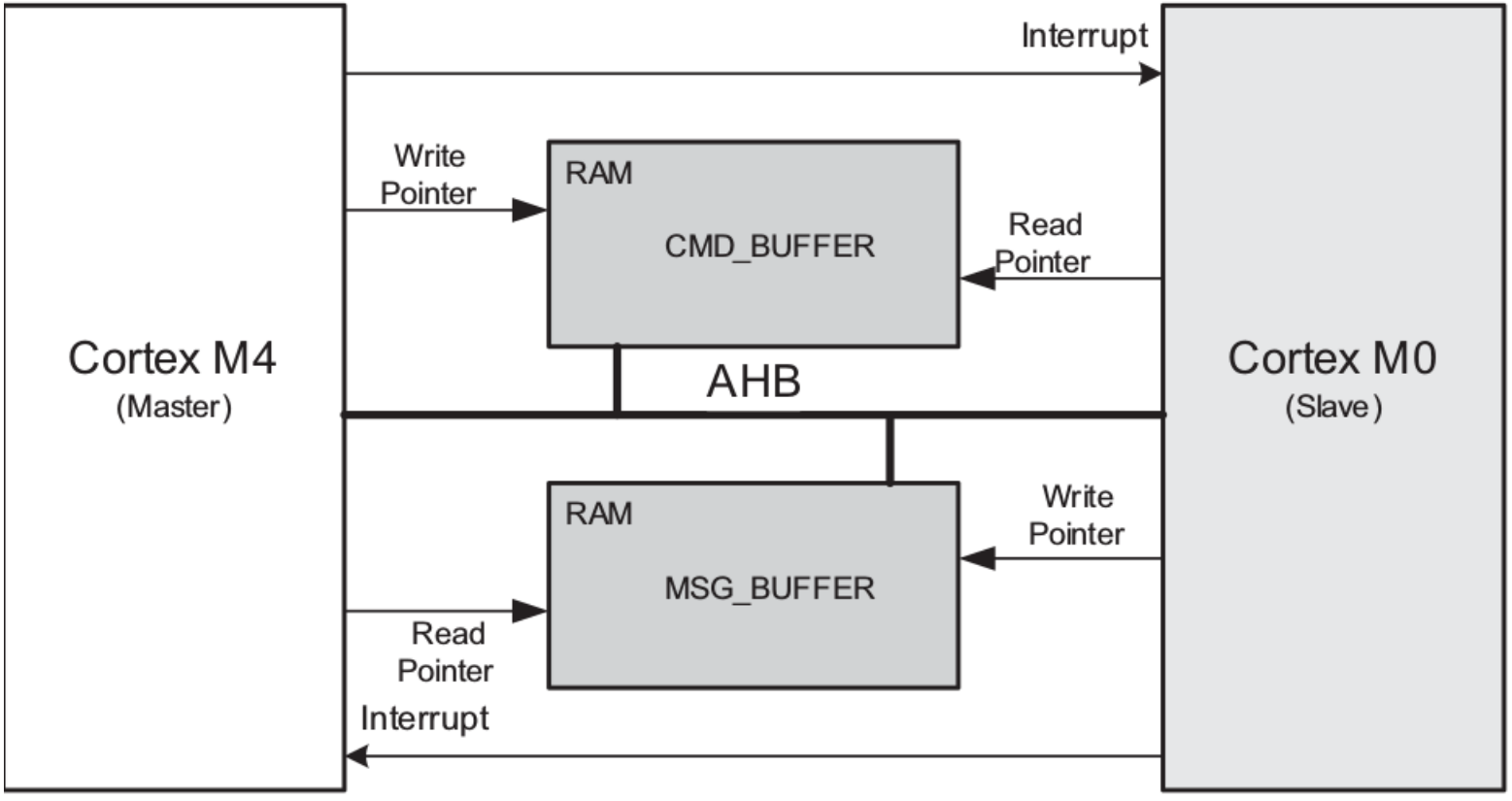
Successfull git branch model



Ramas creadas<+(1)->

- Master
- Develop
- Adquisición
- Almacenamiento
- Interfaz de usuario
- Control
- Ceedling

Inter Process Communications



= M0 subsystem

= M4 subsystem

= shared

Agenda

Motivación

Planificación

Metodología

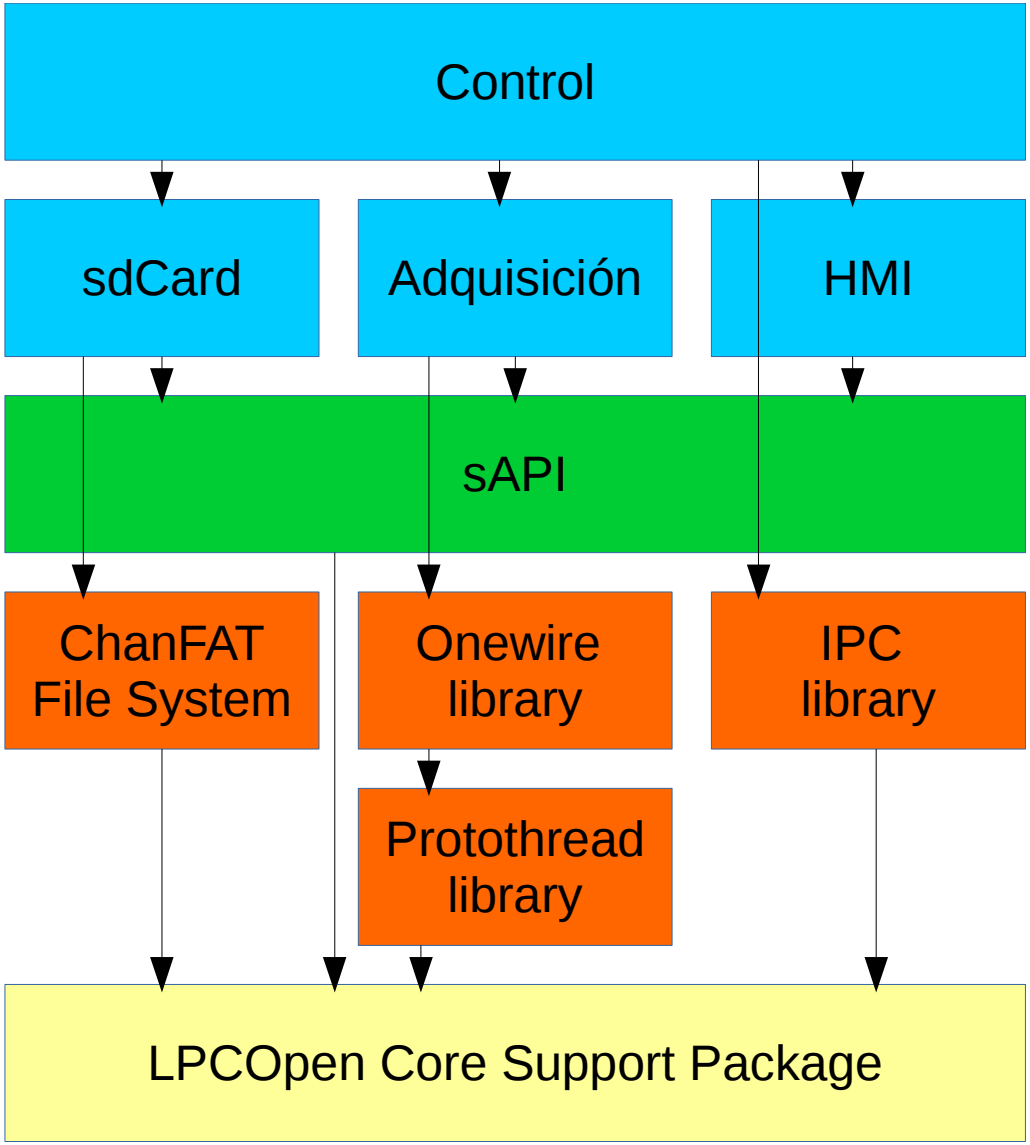
Implementación

Testing

Demo

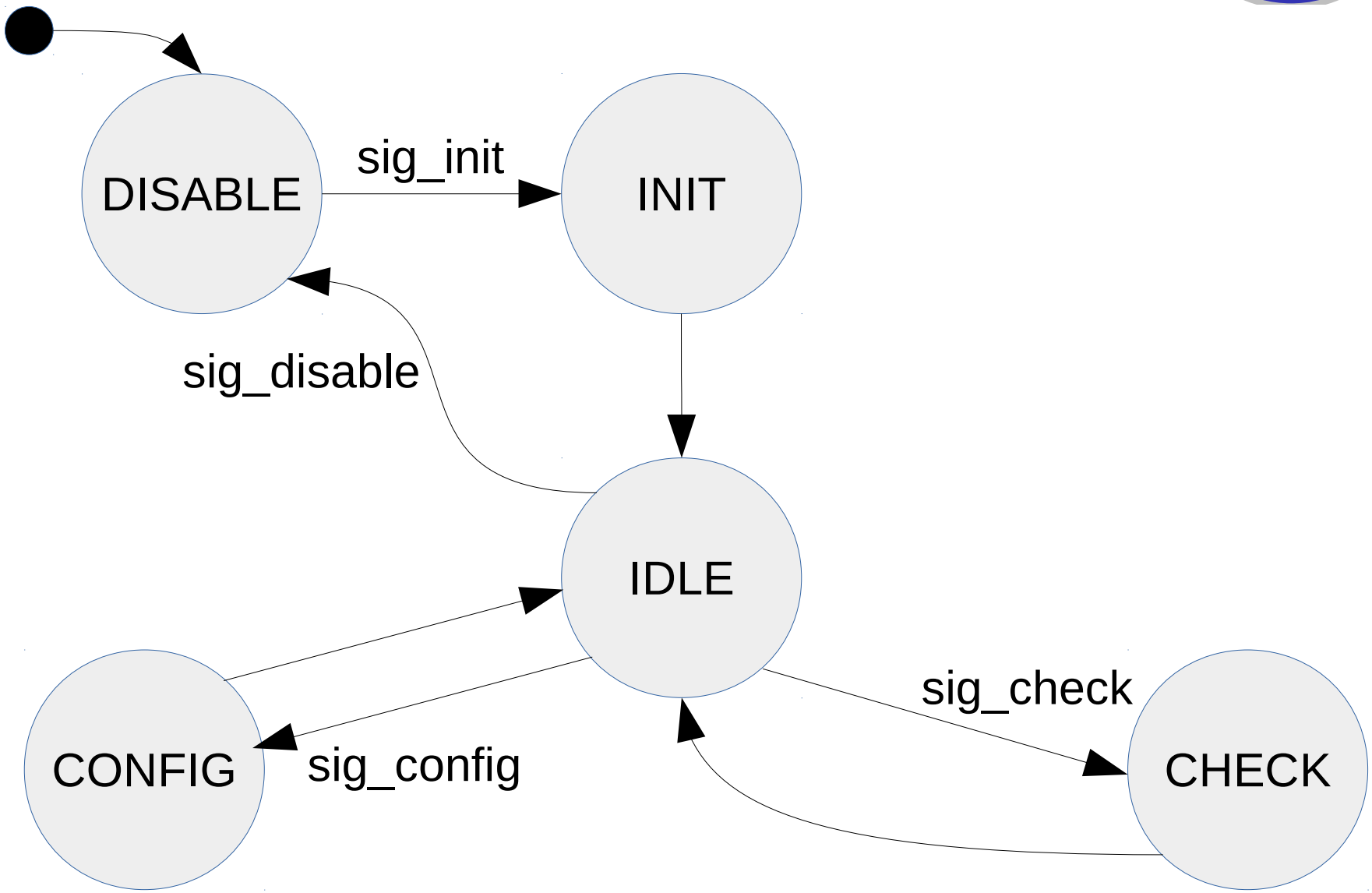
Conclusiones

Modelo de capas



Máquina de Estados Finitos

Módulo genérico



Máquina de Estados Finitos

Módulo genérico

Máquina de Estados Finitos

Módulo genérico

algo

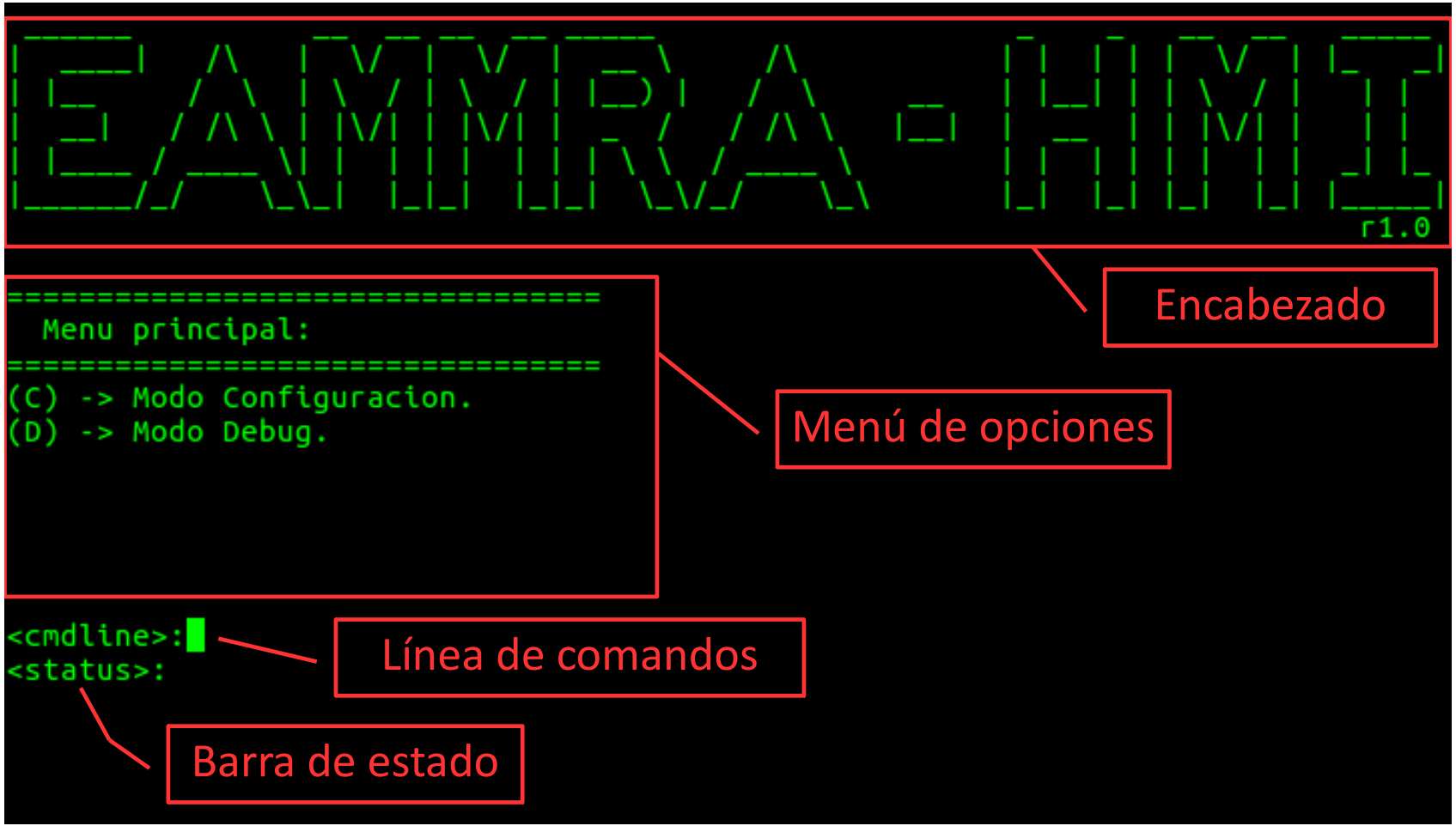
Máquina de Estados Finitos

Módulo genérico

algo

otra

Interfaz de usuario



Agenda

Motivación

Planificación

Metodología

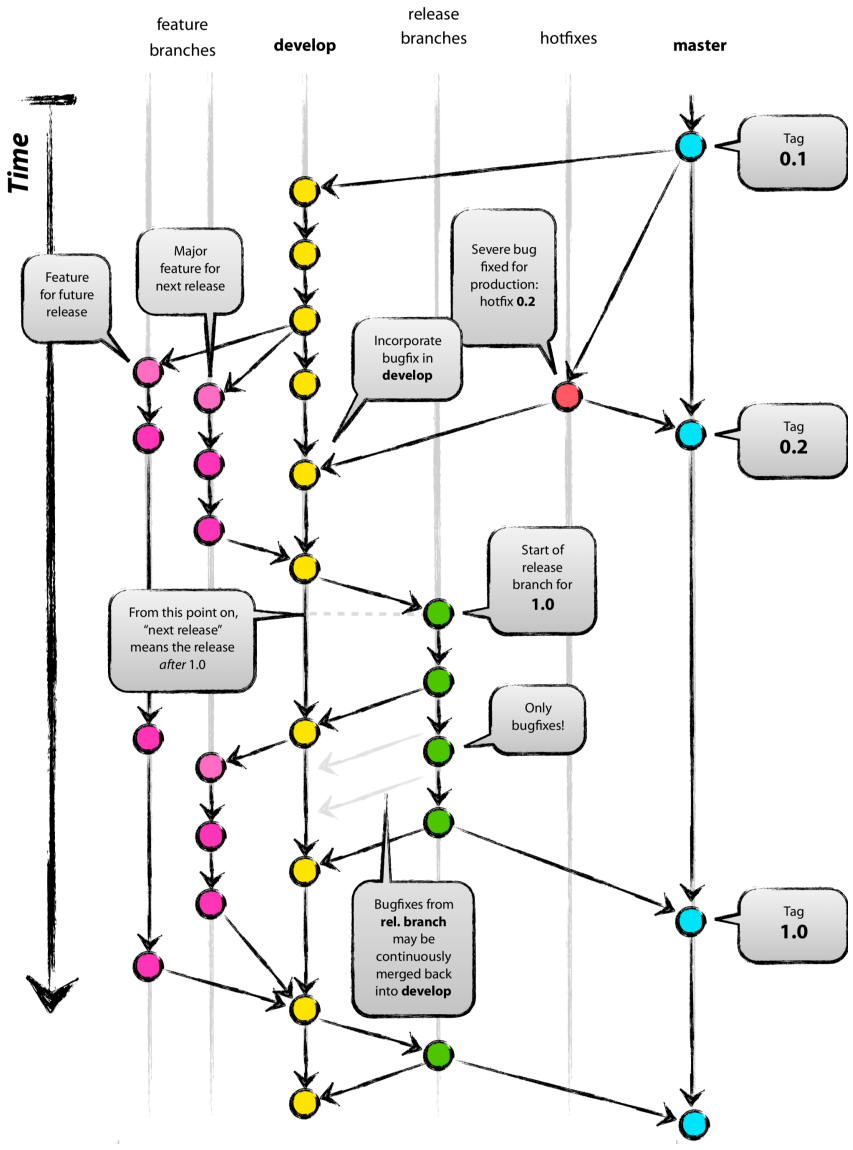
Implementación

Testing

Demo

Conclusiones

Modelo de ramas



Agenda

Motivación

Planificación

Metodología

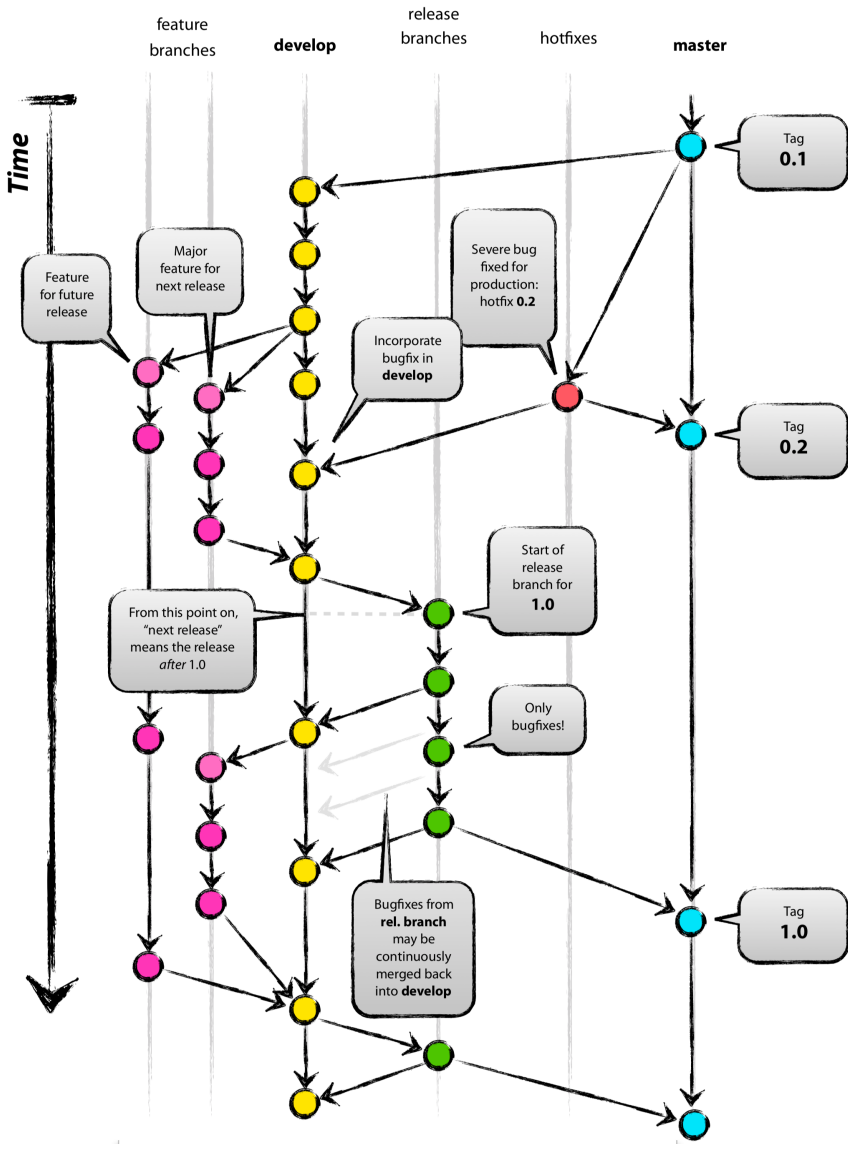
Implementación

Testing

Demo

Conclusiones

Modelo de ramas



Agenda

Motivación

Planificación

Metodología

Implementación

Testing

Demo

Conclusiones

¿Sobre qué hace falta alertar?

Tecnologías utilizadas

¿Preguntas?

Protothreads

Multitasking cooperativo


```
struct pt { unsigned short lc; };

#define PT_THREAD(name_args)  char name_args

#define PT_BEGIN(pt)          switch(pt->lc) { case 0:

#define PT_WAIT_UNTIL(pt, c)  pt->lc = __LINE__; \
                              case __LINE__: \
                              if(!(c)) return 0

#define PT_END(pt)            } pt->lc = 0; return 2

#define PT_INIT(pt)           pt->lc = 0
```

Protothreads

Multitasking cooperativo


```
static
PT_THREAD(example(struct pt
    *pt))
{
  PT_BEGIN(pt);

  while(1) {
    PT_WAIT_UNTIL(pt,
      counter == 1000);
    printf("Threshold
    reached\n");
    counter = 0;
  }

  PT_END(pt);
}
```

Protothreads

```
static
char example(struct pt *pt)
{

  switch(pt->lc) { case 0:

  while(1) {
    pt->lc = 12; case 12:
    if(!(counter == 1000))
    return 0;
    printf("Threshold
    reached\n");
    counter = 0;
  }
  } pt->lc = 0; return 2;
}
```

Traducción