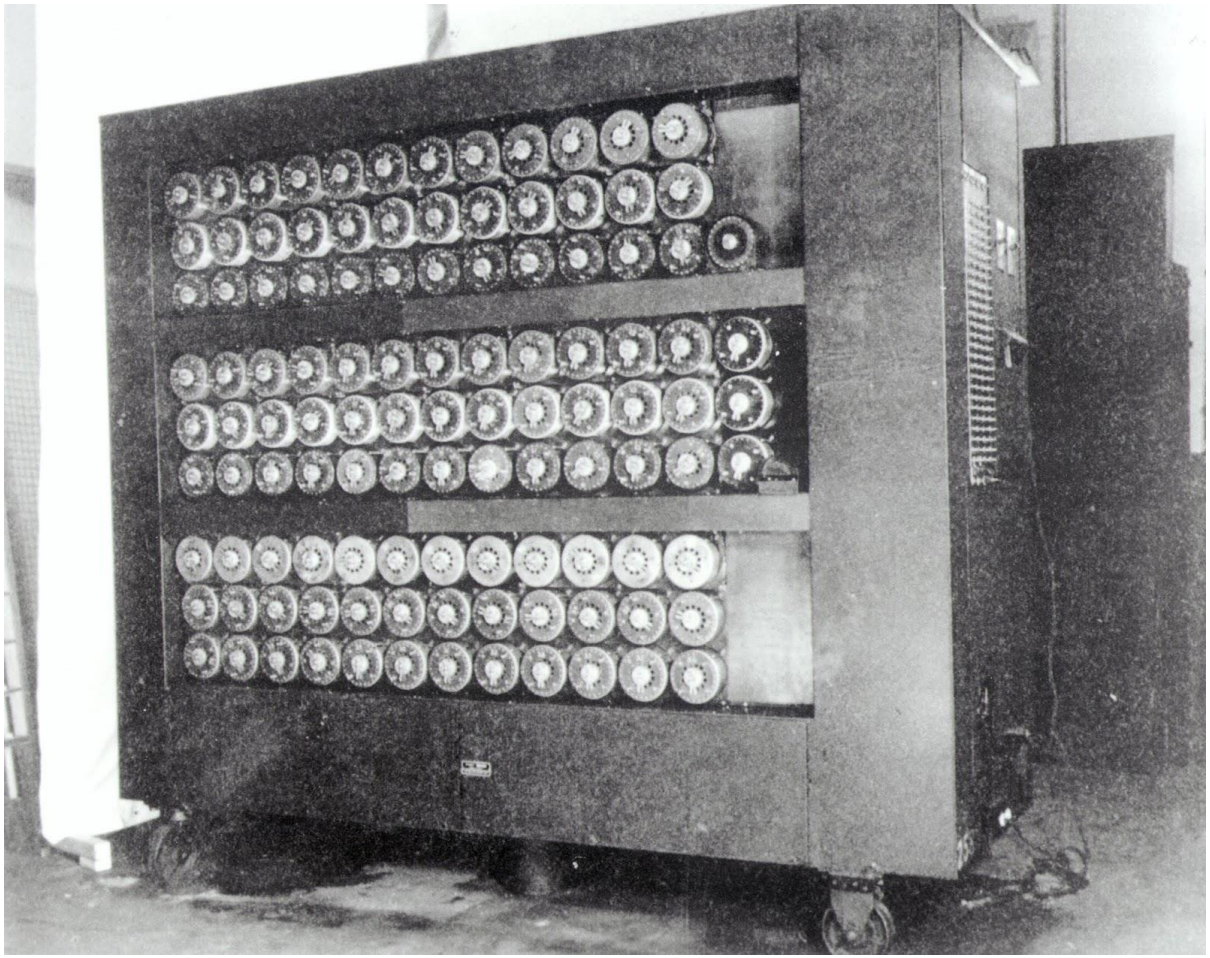
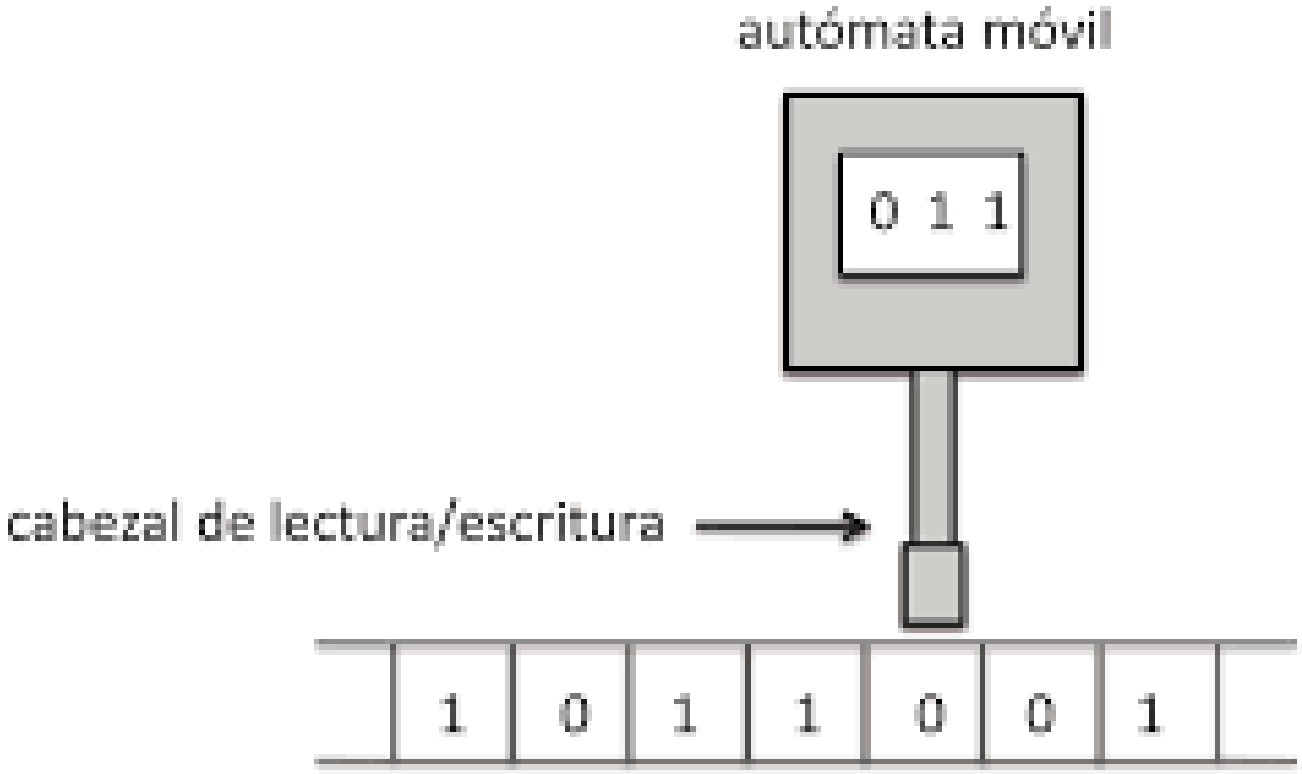
Variability-intensive systems
A pragmatic view

José A. Galindo

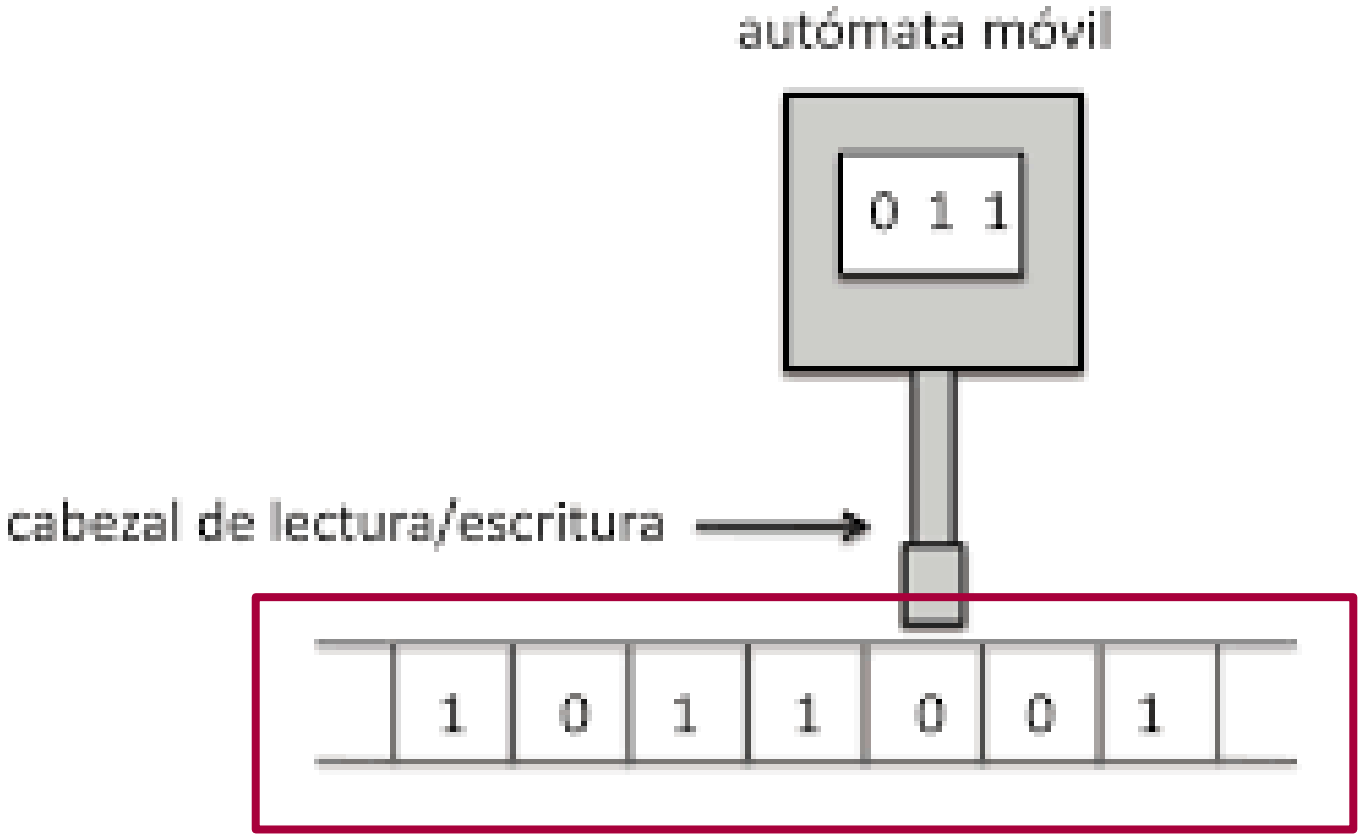
¿Que es un programa de ordenador?



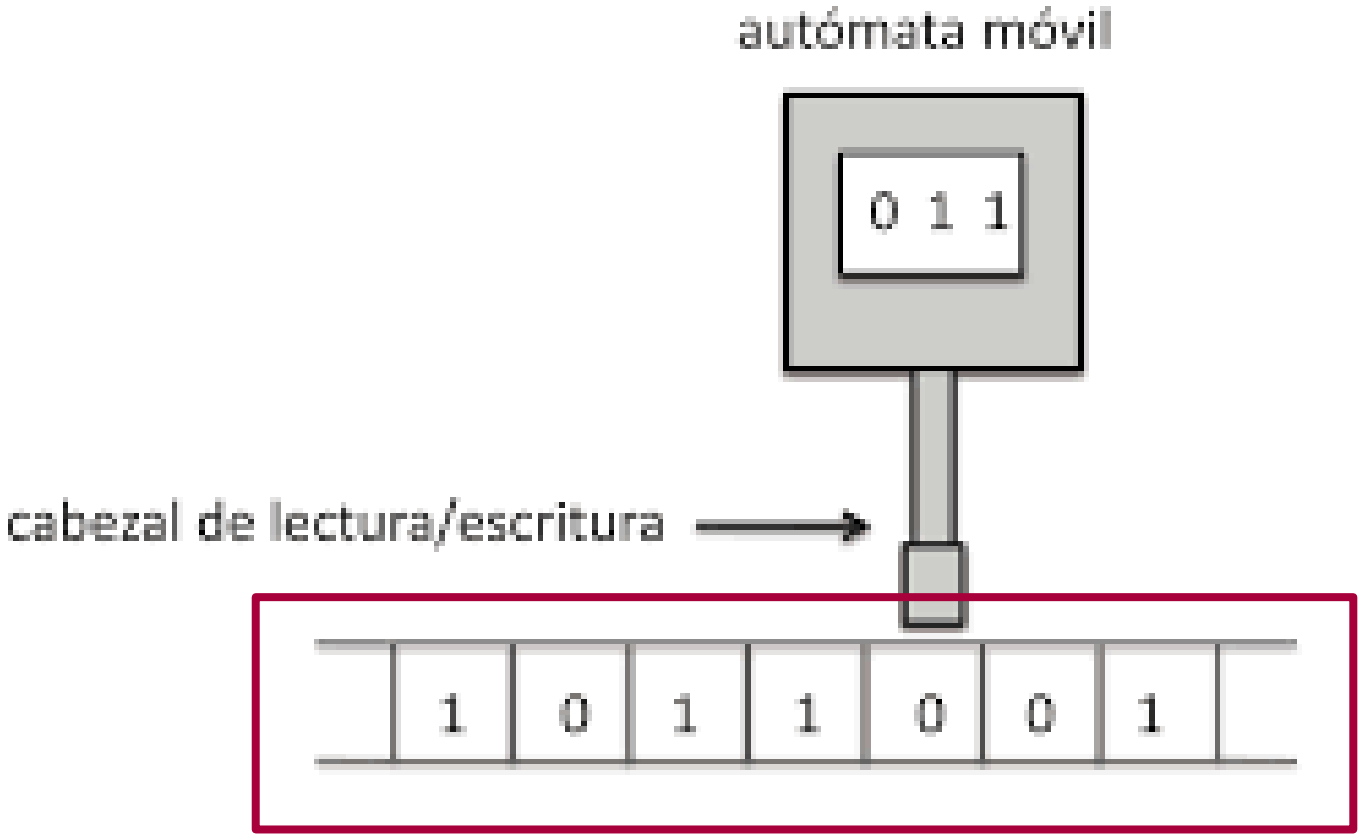
¿Que es un programa de ordenador?



¿Que es un programa de ordenador?



¿Que es un programa de ordenador?

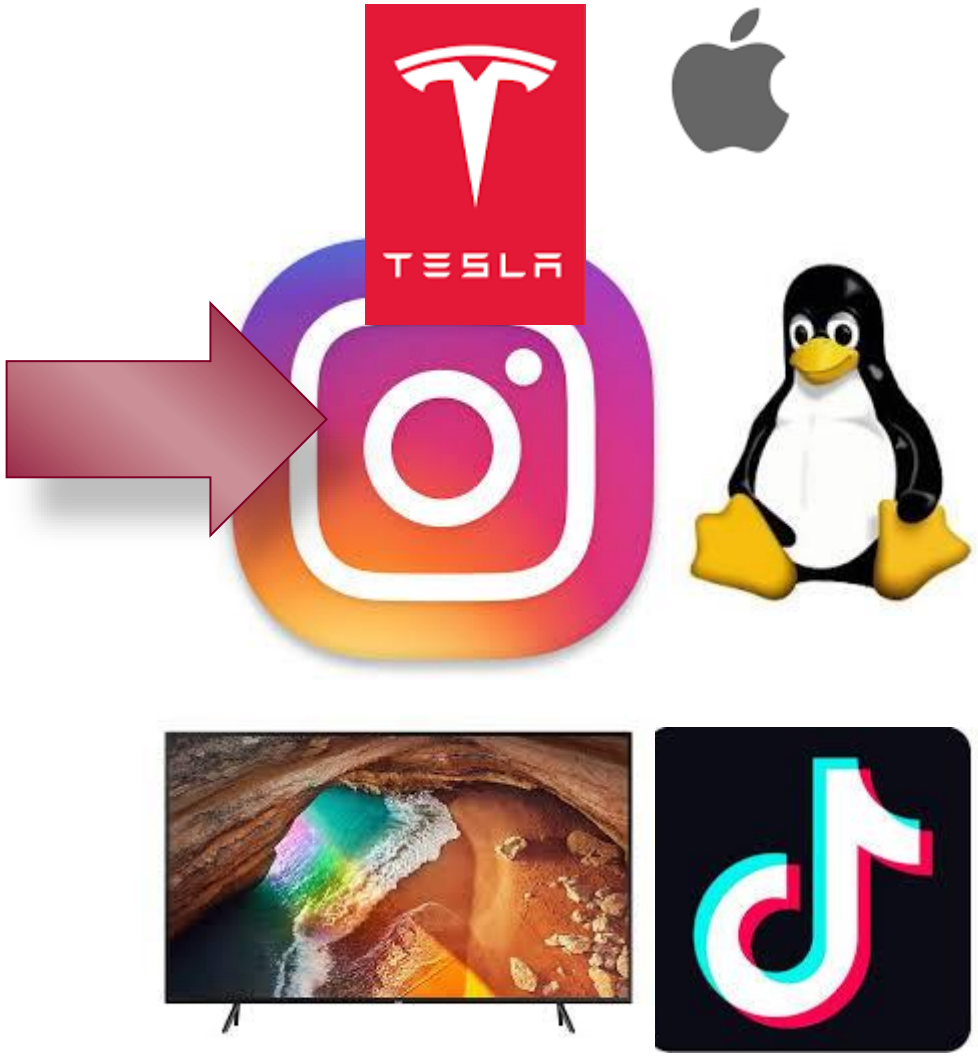


Mi hola mundo implementado en node.js

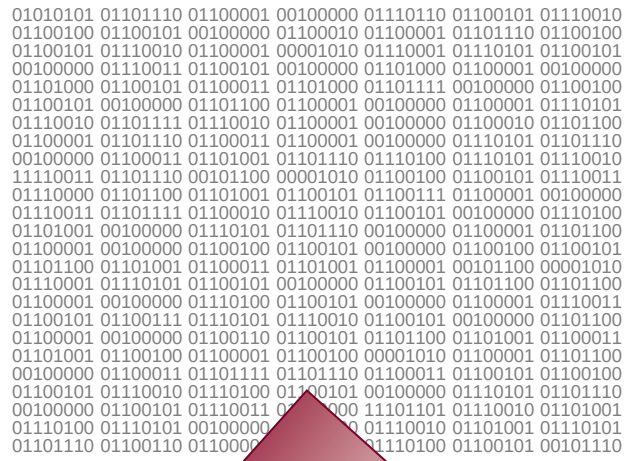
01010101 01101110 01100001 00100000 01110110 01100101 01110010
01100100 01100101 00100000 01100010 01100001 01101110 01100100
01100101 01110010 01100001 00001010 01110001 01110101 01100101
00100000 01110011 01100101 00100000 01101000 01100001 00100000
01101000 01100101 01100011 01101000 01101111 00100000 01100100
01100101 00100000 01101100 01100001 00100000 01100001 01110101
01110010 01101111 01110010 01100001 00100000 01100010 01101100
01100001 01101110 01100011 01100001 00100000 01110101 01101110
00100000 01100011 01101001 01101110 01110100 01110101 01110010
11110011 01101110 00101100 00001010 01100100 01100101 01110011
01110000 01101100 01101001 01100101 01100111 01100001 00100000
01110011 01101111 01100010 01110010 01100101 00100000 01110100
01101001 00100000 01110101 01101110 00100000 01100001 01101100
01100001 00100000 01100100 01100101 00100000 01100100 01100101
01101100 01101001 01100011 01101001 01100001 00101100 00001010
01110001 01110101 01100101 00100000 01100101 01101100 01101100
01100001 00100000 01110100 01100101 00100000 01100001 01110011
01100101 01100111 01110101 01110010 01100101 00100000 01101100
01100001 00100000 01100110 01100101 01101100 01101001 01100011
01101001 01100100 01100001 01100100 00001010 01100001 01101100
00100000 01100011 01101111 01101110 01100011 01100101 01100100
01100101 01110010 01110100 01100101 00100000 01110101 01101110
00100000 01100101 01110011 01110000 11101101 01110010 01101001
01110100 01110101 00100000 01110100 01110010 01101001 01110101
01101110 01100110 01100001 01101110 01110100 01100101 00101110

¿Por qué ya no programamos en ensamblador?

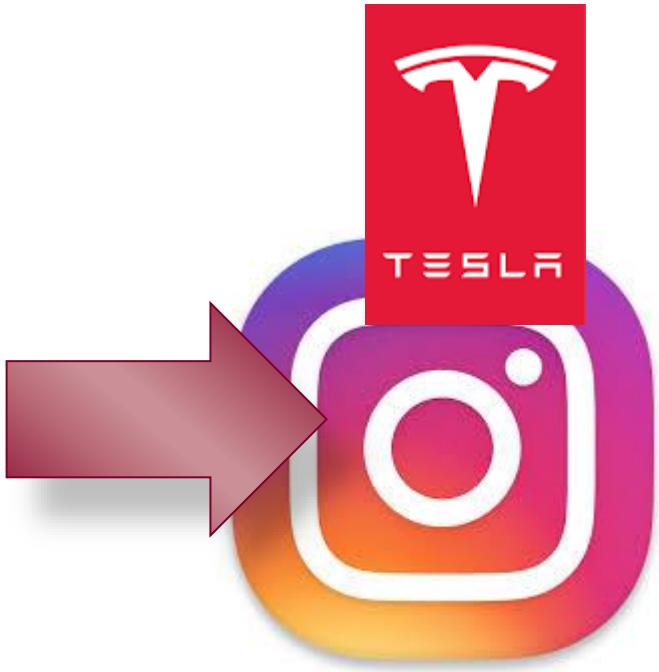
```
01010101 01101110 01100001 00100000 01110110
01100101 01110010 01100100 01100101 00100000
01100010 01100001 01101110 01100100 01100101
01110010 01100001 00001010 01110001 01110101
01100101 00100000 01110011 01100101 00100000
01101000 01100001 00100000 01101000 01100101
01100011 01101000 01101111 00100000 01100100
01100101 00100000 01101100 01100001 00100000
01100001 01110101 01110010 01101111 01110010
01100001 00100000 01100010 01101100 01100001
01101110 01100011 01100001 00100000 01110101
01101110 00100000 01100011 01101001 01101110
01110100 01110101 01110010 11110011 01101110
00101100 00001010 01100100 01100101 01110011
01110000 01101100 01101001 01100101 01100111
01100001 00100000 01110011 01101111 01100010
01110010 01100101 00100000 01110100 01101001
00100000 01110101 01101110 00100000 01100001
01101100 01100001 00100000 01100100 01100101
00100000 01100100 01100101 01101100 01101001
01100011 01101001 01100001 00101100 00001010
01110001 01110101 01100101 00100000 01100101
01101100 01101100 01100001 00100000 01110100
01100101 00100000 01100001 01110011 01100101
01100111 01110101 01110010 01100101 00100000
01101100 01100001 00100000 01100110 01100101
01101100 01101001 01100011 01101001 01100100
01100001 01100100 00001010 01100001 01101100
00100000 01100011 01101111 01101110 01100011
01100101 01100100 01100101 01110010 01110100
01100101 00100000 01110101 01101110 00100000
01100101 01110011 01110000 11101101 01110010
01101001 01110100 01110101 00100000 01110100
01110010 01101001 01110101 01101110 01100110
01100001 01101110 01110100 01100101 00101110
```



Necesitamos estructuras simples para crear sistemas complejos

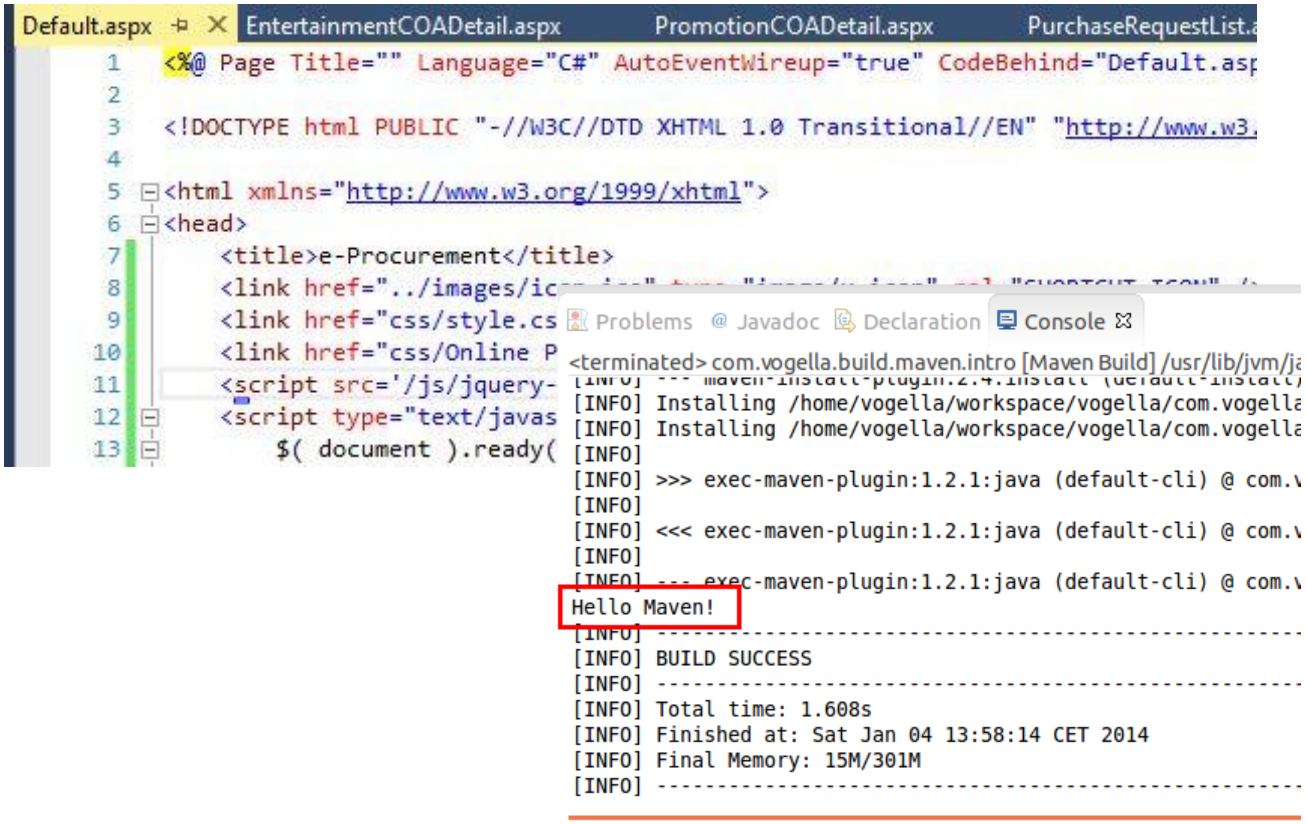


Software Engenieri ng

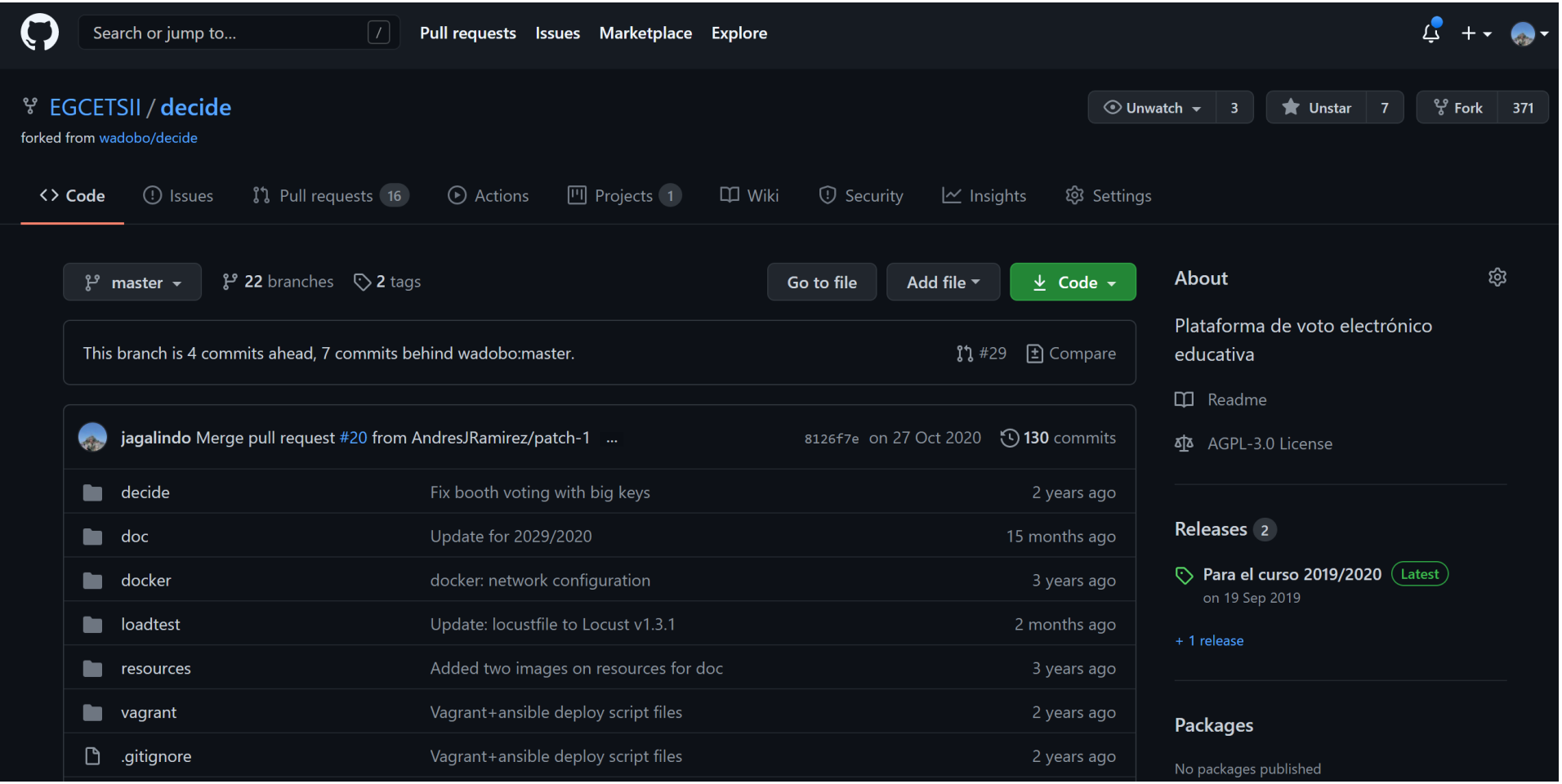


```
void bubblesort(int [ ... ] length)
{
        // Bubble largest number toward the right
        for (int i = length-1; i > 0; i--)
                for (int j = 0; j < i; j++)
                        if (arr[j] > arr[j+1])
                        {
                                // Swap the numbers
                                int temp = arr[j+1];
                                arr[j+1] = arr[j];
                                arr[j] = temp;
                        }
}
```

Reuso en el código



Clonando/Forkeando



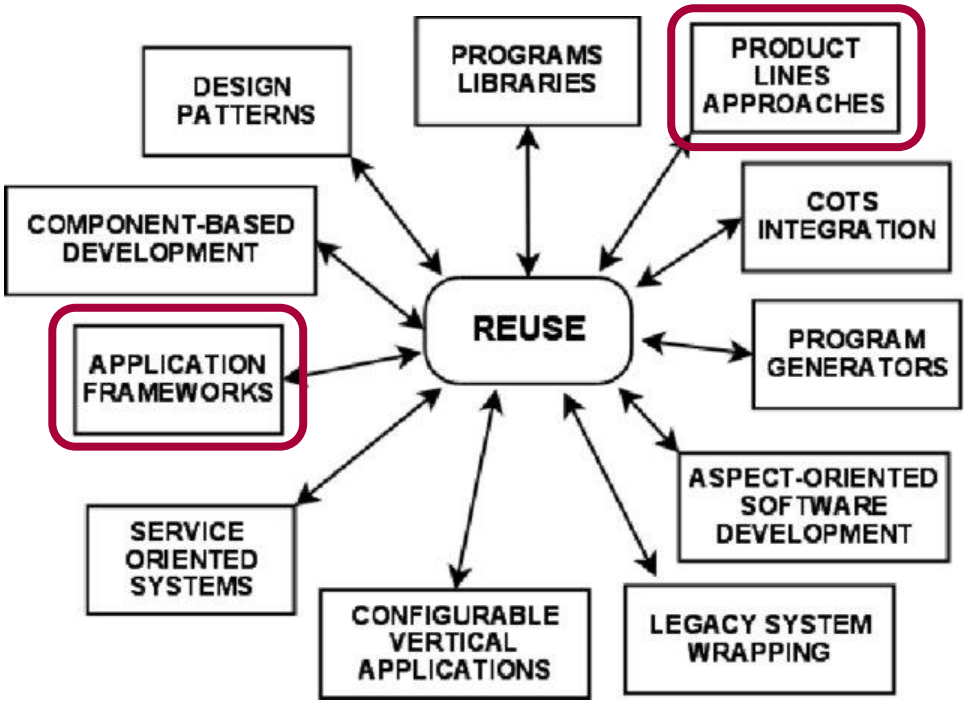
Patrones como base para el reuso

Programando ya reusamos!

```
while( ax < bx)
    ax = ax + 1;
```

This is a possible implementation:

```
top: cmp ax,bx              ; check loop condition
     jae next               ; false? exit loop
     inc ax                 ; body of loop
     jmp top                ; repeat the loop
next:
```

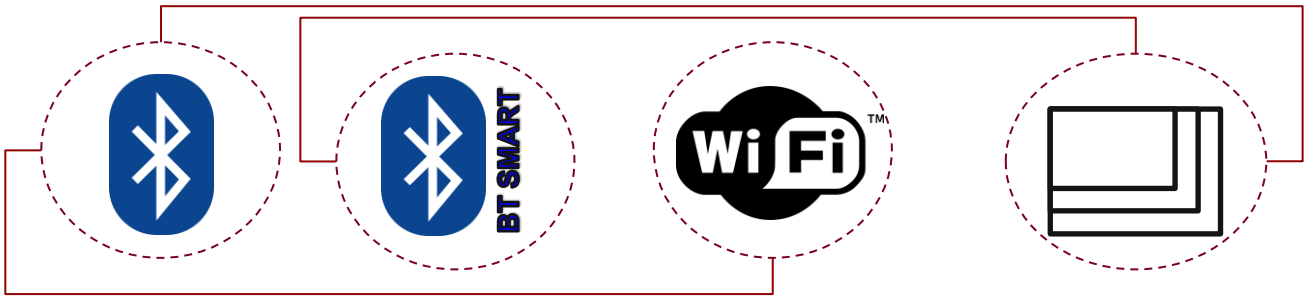


[RiSE Reference Model for Software Reuse Adoption in Brazilian Companies]

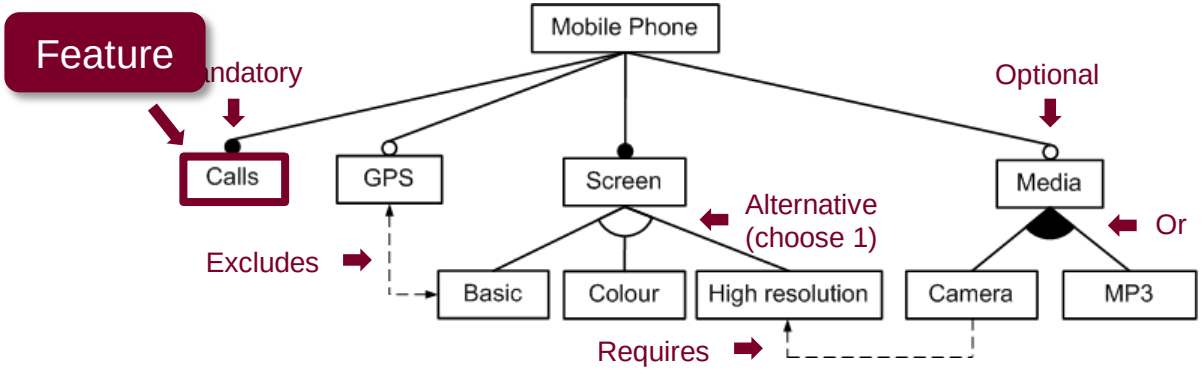
Abstracción

Variability Intensive Systems

Variability Modeling



Feature Models



Variability Models

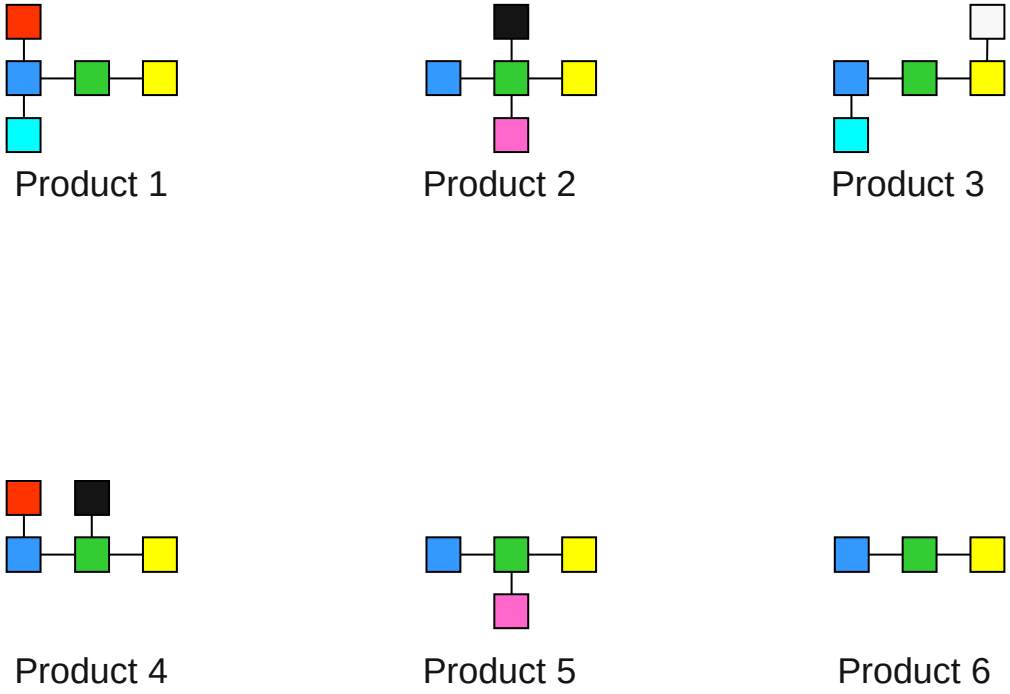
Ventajas y problemas

- Encapsular funcionalidad permite tener un catalogo de productos amplios
- No es fácil encapsular funcionalidad "ortogonal"
  - Requisitos no funcionales (e.g. grado de seguridad de la app)
  - Requisitos funcionales difusos (e.g. idiomas)
  - Require de composición ya sea en tiempo de ejecución o de compilación

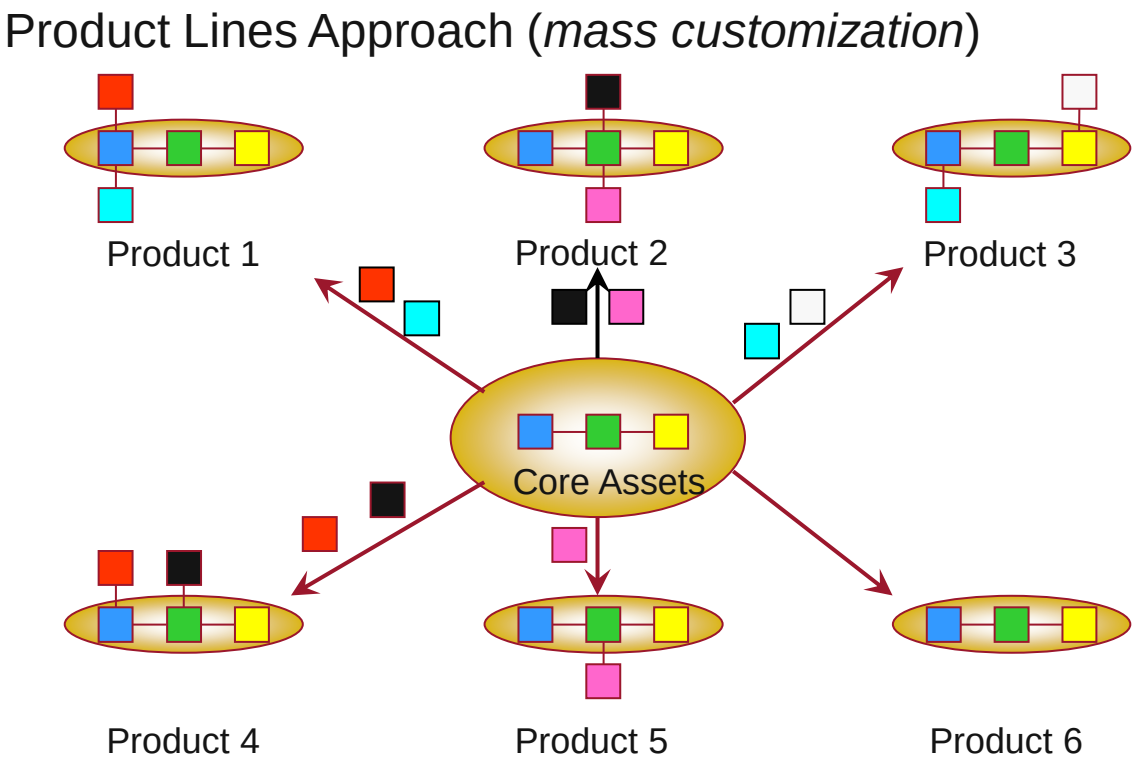
But, how to develop my SPL

Software product lines

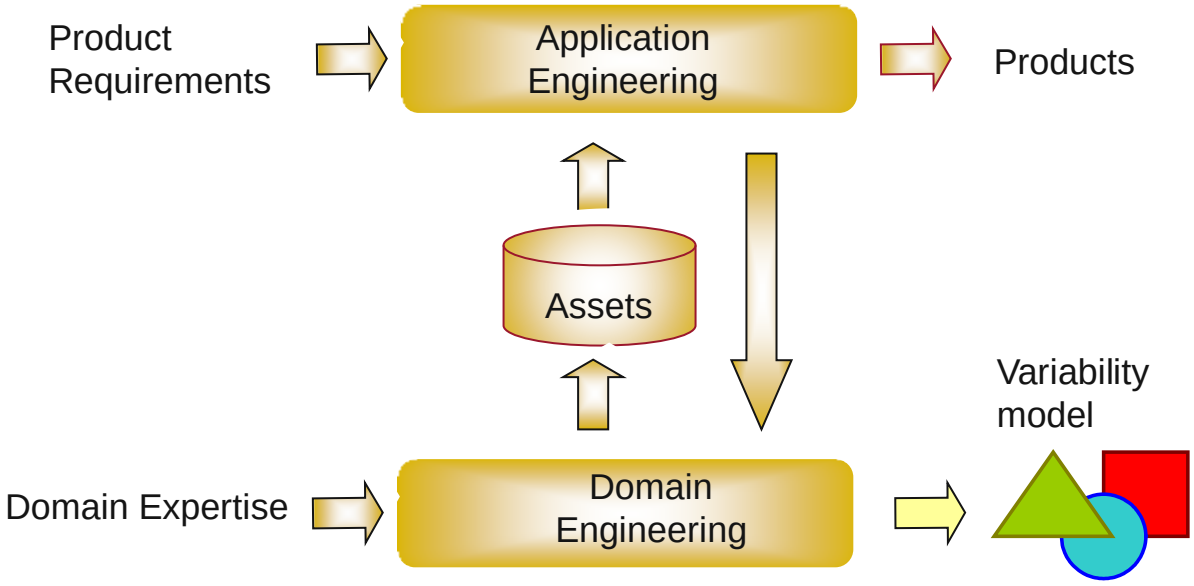
Traditional Approach (*mass production*)



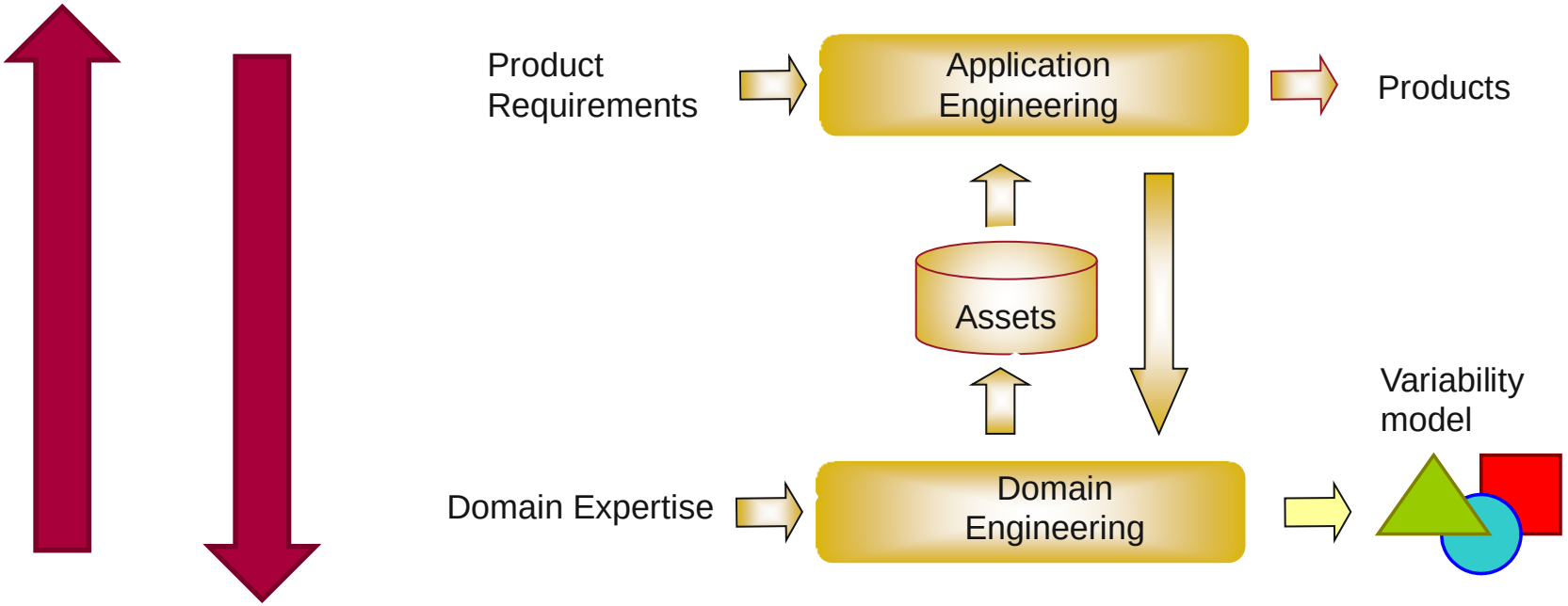
Software product lines

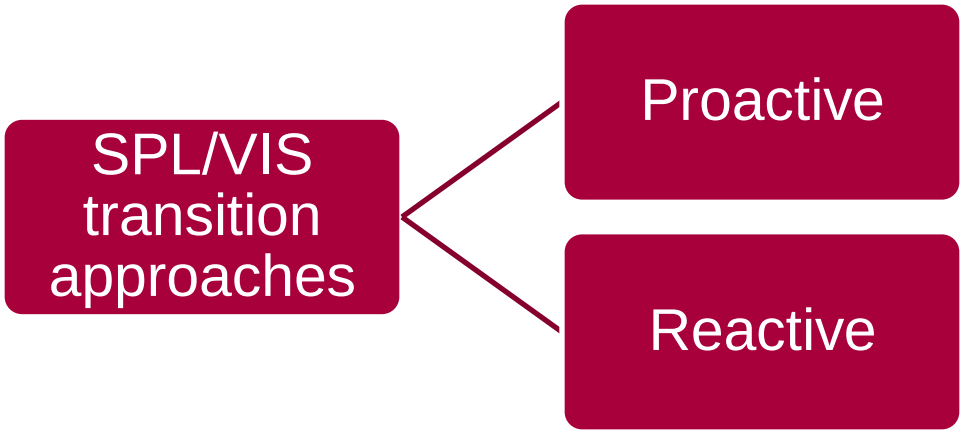


SPL: Activities

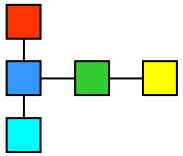


Top-down vs bottom-up

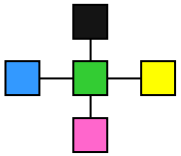



SPL/VIS transition approaches
Proactive
Reactive

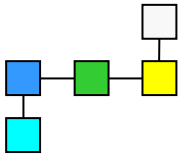
Reactive approach



Product 1

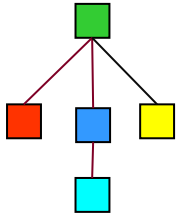


Product 2



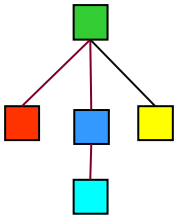
Product 3

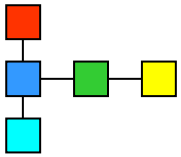
Products



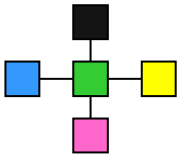
FM

Reactive approach

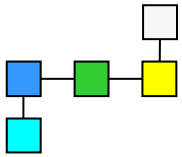




Product 1



Product 2



Product 3

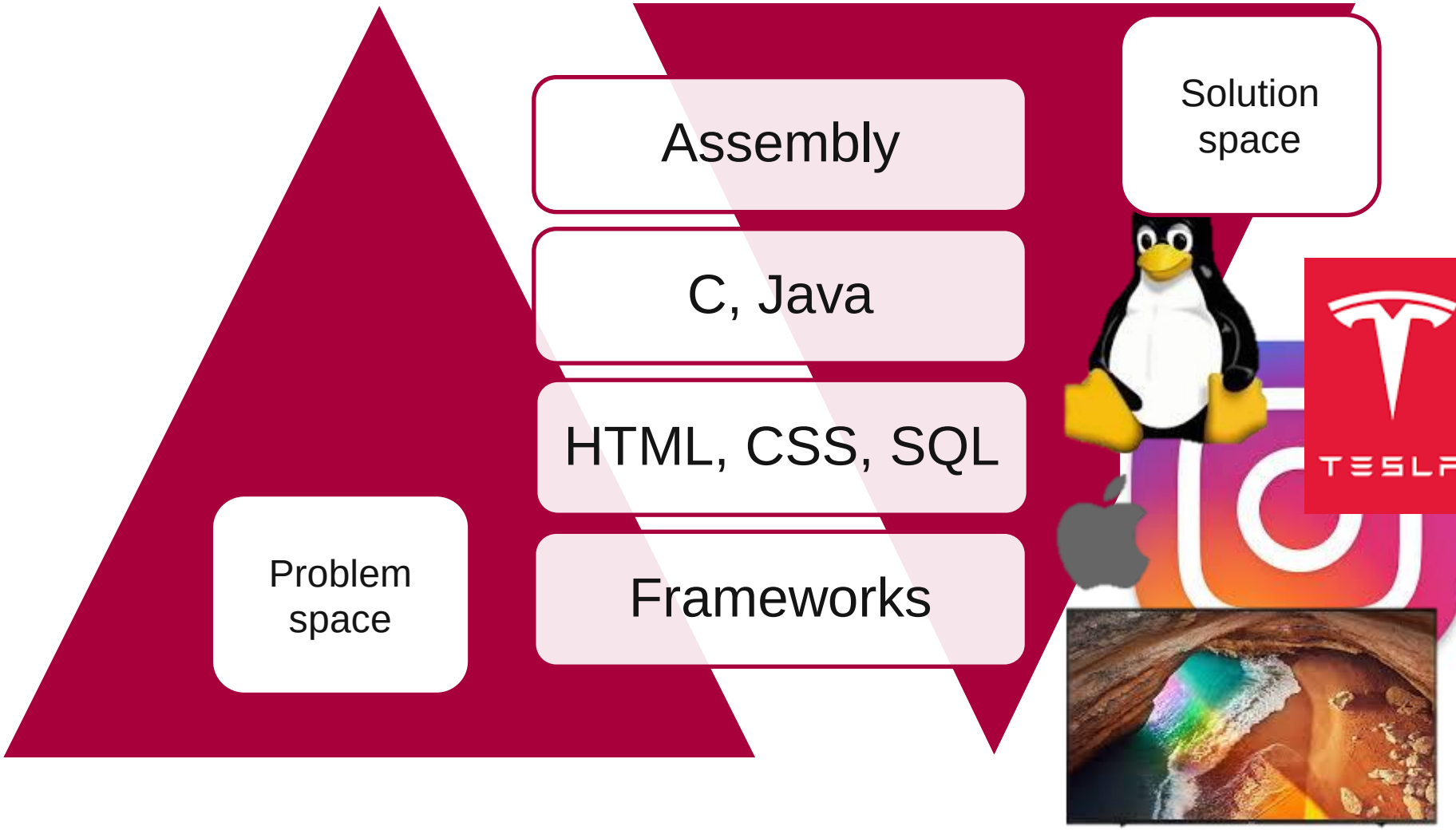
How to develop SPL proactively

- But how?
  - Templates
  - Compilación
  - Preprocesamiento
  - Cargadores de clases
  - Modularización

Frameworks

- Framework son programas, componentes, utilidades que facilitan el desarrollo de aplicaciones de software relativamente estandarizadas, o bien algunas partes de éstas, proporcionando el diseño y, a veces, sus componentes auxiliares como librerías o herramientas.

Abstraction Gap



Bunch of frameworks

Frameworks are VIS

Black and white boxes

How frameworks compare to other object-oriented reuse techniques.

FRAMEWORKS = (COMPONENTS + PATTERNS)

Frameworks are an object-oriented reuse technique. They share many characteristics with reuse techniques in general [8], and object-oriented reuse techniques in particular. Although they have been used successfully for some time, and are an important part of the culture of long-time object-oriented developers, they are not well understood outside the object-oriented community and are often misused. Moreover, there is confusion about whether frameworks are large-scale patterns, or whether they are just another kind of component.

https://dl.acm.org/doi/pdf/10.1145/262793.262799

El principio "Hollywood"

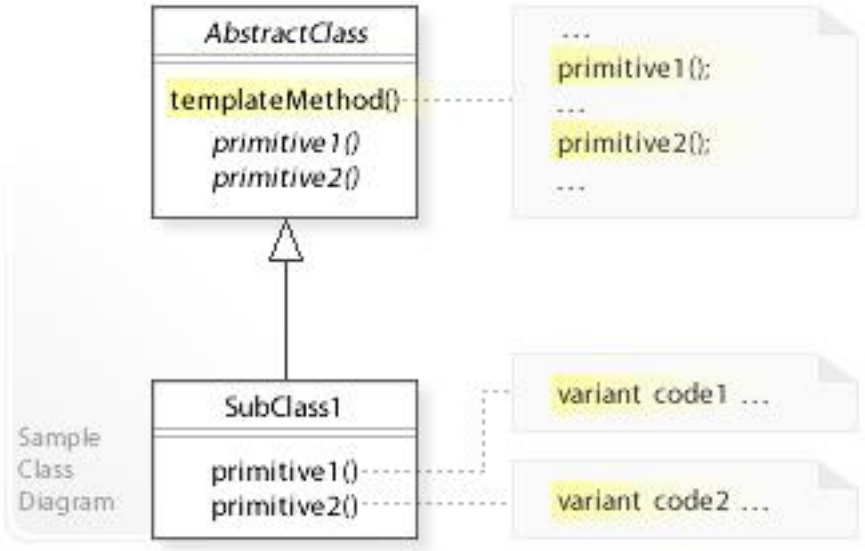
"Don't call us, we'll call you."

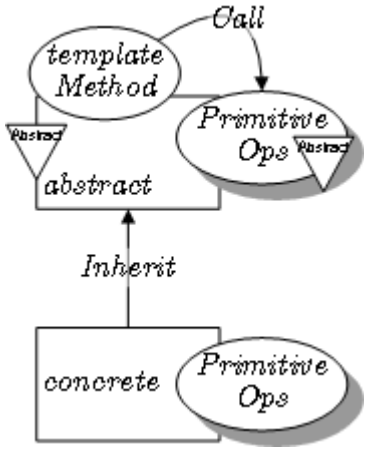
```
public class ServerFacade {
    public <K, V> V respondToRequest(K request, DAO dao) {
        return dao.getData(request);
    }
}
```

Larman, C (2001), Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (2nd ed.), Prentice Hall, ISBN 978-0-13-092569-5

Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John (1994). Design Patterns. Addison-Wesley. ISBN 0-201-63361-2.

El patrón plantilla





¿Qué elementos tiene un framework?

- **RANURAS O HOT SPOTS:** Puntos en los que variarán las diferentes aplicaciones que creemos en el mismo dominio del problema.

- **METODOS DE ANCLAJE o HOOK METHODS:** constituyen las operaciones a las que el usuario tendrá acceso para la utilización de las clases concretas aportadas por el framework.

- **PUNTOS CONGELADOS o CONTRATOS DEL FRAMEWORK o FROZEN SPOTS:** Elementos inmutables que pertenecen a la estructura interna o esqueleto del marco.

Hot spots (Extensiones)

- Son clases o métodos abstractos que deben ser ejecutados o puestos en ejecución. Puntos en los que variarán las diferentes aplicaciones que creemos dentro del mismo dominio del problema.
- Estos lugares indican donde puede extenderse el framework por los usuarios del mismo.
- Estructuralmente, un hot spot se compone de una clase base abstracta, de diversas clases derivadas y de posibles clases adicionales con sus correspondientes relaciones.
- Un framework puede tener varios hot spots pero la elección de uno de ellos puede restringir la elección de otros.

Ejemplos

```
@SpringBootApplication
@RestController
public class SocialApplication extends WebSecurityConfigurerAdapter {

    // ...

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        // @formatter:off
        http
            .authorizeRequests(a -> a
                .antMatchers("/", "/error", "/webjars/**").permitAll()
                .anyRequest().authenticated()
            )
            .exceptionHandling(e -> e
                .authenticationEntryPoint(new
HttpStatusEntryPoint(HttpStatus.UNAUTHORIZED))
            )
            .oauth2Login();
        // @formatter:on
    }

}
```

https://spring.io/guides/tutorials/spring-boot-oauth2/

Hooks

- Elementos públicos de los frameworks que representan recursos sobre los que los clientes pueden construir, métodos que, una vez definidos, conformarán la distinción entre varias aplicaciones del mismo dominio del problema. Constituyen las operaciones a los que el usuario tendrá acceso, para la utilización de las clases concretas aportadas por el framework.

Ejemplos

COPY

```
package com.example.restservice;

import java.util.concurrent.atomic.AtomicLong;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class GreetingController {

        private static final String template = "Hello, %s!";
        private final AtomicLong counter = new AtomicLong();

        @GetMapping("/greeting")
        public Greeting greeting(@RequestParam(value = "name", defaultValue =
"World") String name) {
                return new Greeting(counter.incrementAndGet(),
String.format(template, name));
        }
}
```

https://spring.io/guides/gs/rest-service/

FROZEN SPOTS

- Elementos inmutables, pertenecientes a la estructura interna o esqueleto. Código puesto en ejecución que llamarán a uno o más puntos calientes proporcionados por el ejecutor. Constituirán por tanto el núcleo, infraestructura y diseño interno del marco de trabajo.

- Sirven para capturar por tanto la funcionalidad común dentro del dominio del problema. Formalizan las partes del framework reutilizables. En este caso, aplican una estructura encapsulada a la que no tendrá acceso el usuario.

Tipos de Frameworks

- **FRAMEWOKS DE CAJA BLANCA:** orientación a objetos. Basados en la herencia.
- **FRAMEWORKS DE CAJA NEGRA:** soportan extensibilidad
- **FRAMEWORKS DE CAJA GRIS:** es una mezcla de las 2 anteriores, permitiendo la extensión mediante herencia y ligadura dinámica, pero encapsulando la información interna del framework.

Desarrollo de un Framework

- Etapa 1: Análisis
- Etapa 2: Diseño.
- Etapa 3: Instanciación

Análisis

- Estudio del dominio.
  - Viabilidad
  - Nos beneficiará su coste en el medio plazo
  - Que requisitos tiene a día de hoy y que requisitos podrá tener
    - Aprender de experiencias pasadas.

¡¡IMPORTANTE CONOCER ABSTRACCIONES DEL DOMINIO!!

Diseño y desarrollo

- Identificar abstracciones
- Desarrollar algunas aplicaciones que usen el framework
- Comenzaremos diseñando el núcleo y posteriormente los plugins → Ayuda a identificar Hot spots y hooks

Instanciación

Va a variar dependiendo de si es caja blanca o caja negra o si los usuarios pertenecen a nuestra organización o a una externa

Clases que hereden

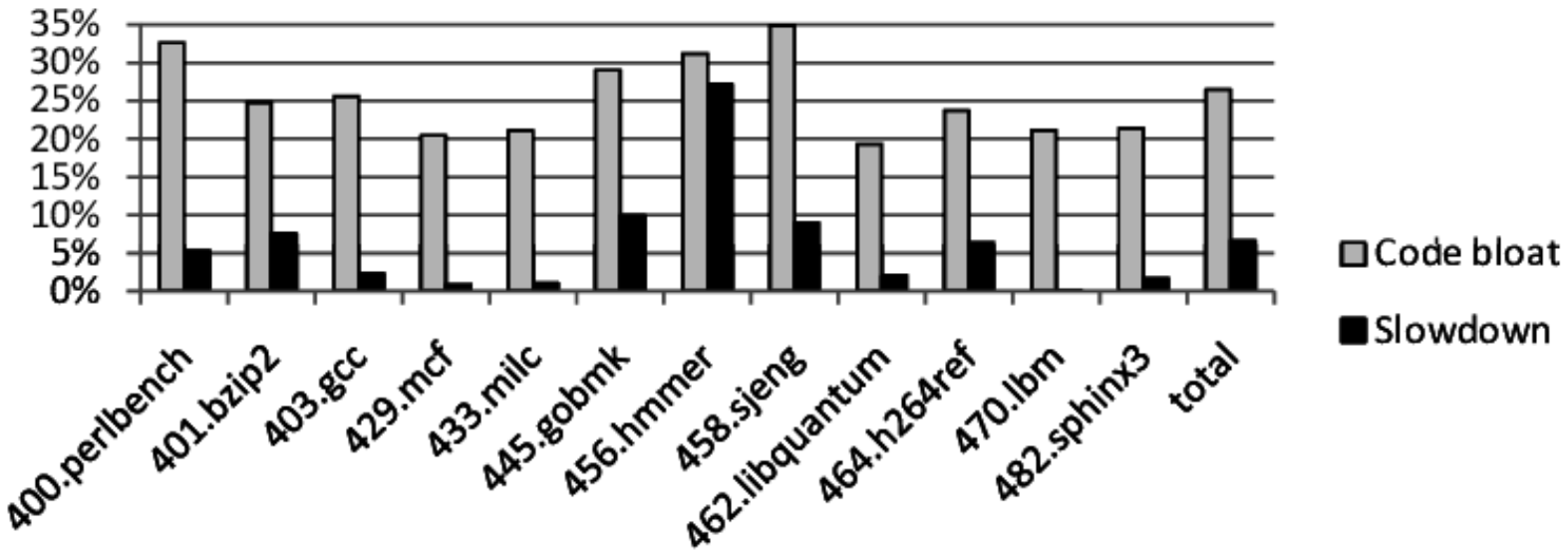
Scripts de configuración y plugins

The Good, the bad and the ugly

The Good, the bad and the ugly

Bloat code / Código inflado

The Good, the bad and the ugly



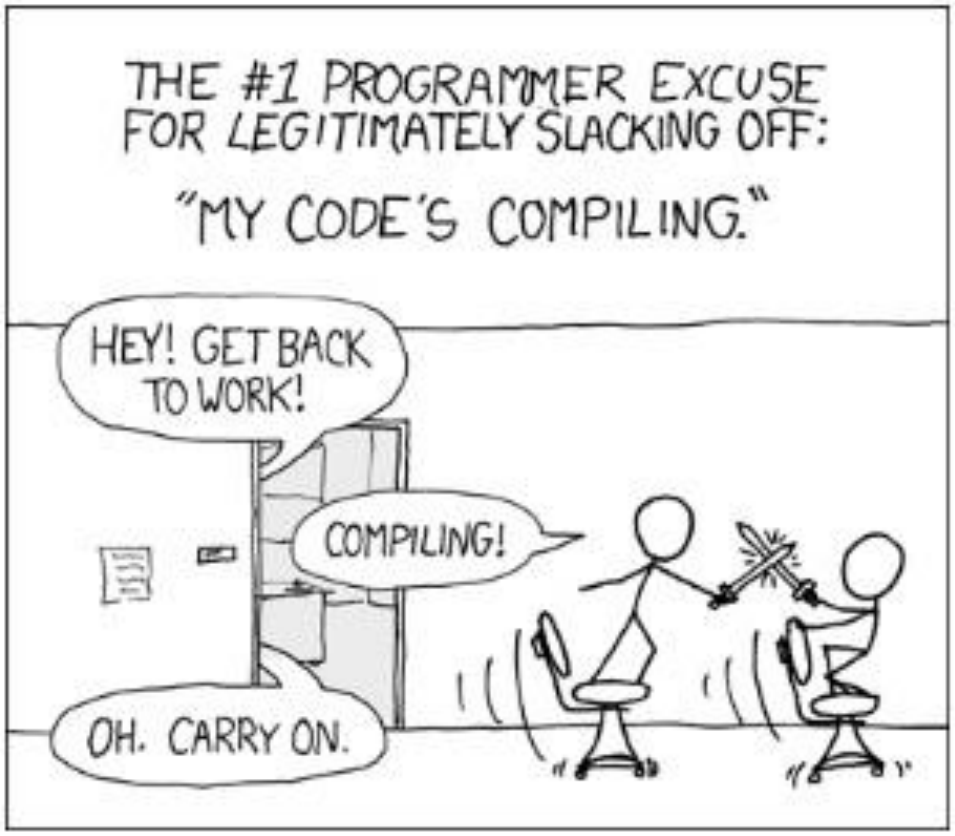
Como conocer el universo de implementaciones

Conocimiento no unificado

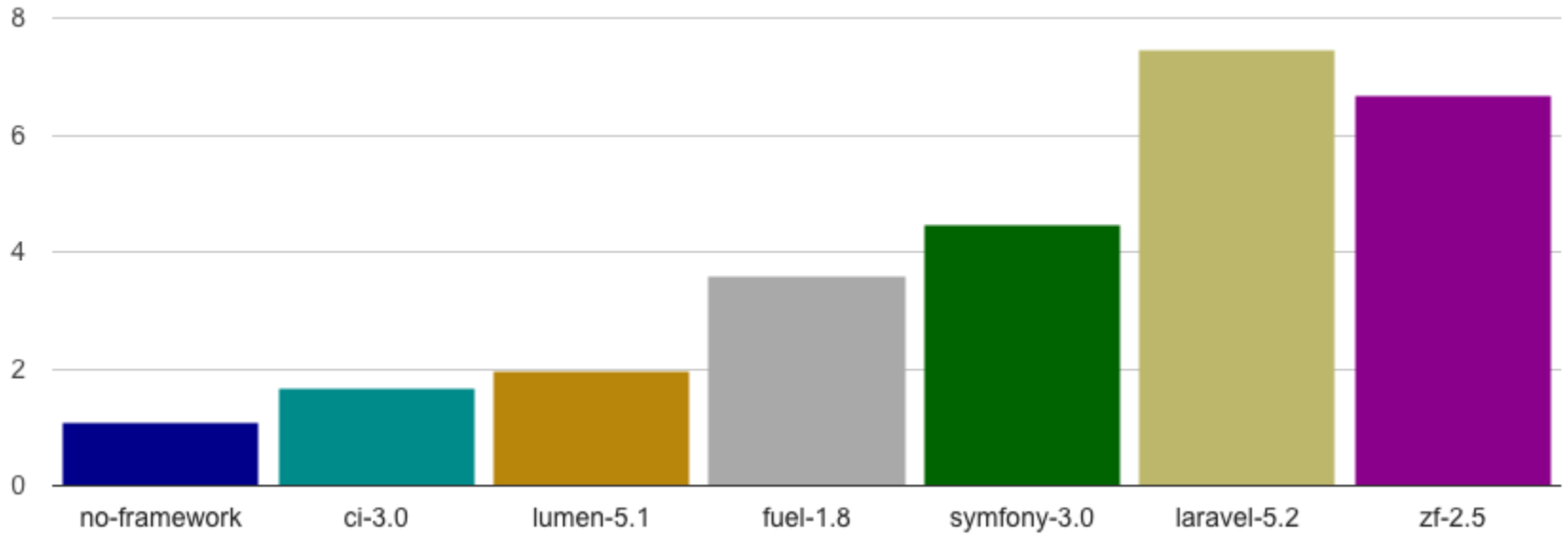
- WooCommerce
  - 19 vulnerabvulnerabilidades a fecha de 2014
- Yoast SEO
  - 10 vulnerabilidades a fecha de 2019
- ZooEffect Plugin for Video player, Photo Gallery Slideshow jQuery and audio / music / podcast – HTML5
  - ZooEffect 1.08 - HTTP Referer Reflected XSS

https://wpscan.com/

Compilation time



Frameworks vs no frameworks (comp+exec)



https://medium.com/@asked_io/php-mvc-framework-showdown-7-0-performance-ba0ed09c1f54

The goods

Promises of Frameworks

Más abstracción

Evita la redundancia

Delimita mejor los problemas

Usa elementos del dominio para programar

Resumen

- Hemos visto
  - Que son los sistemas de alta variabilidad
  - Que tipos de reuso existen
  - Vínculos entre SPL y frameworks (bottom up, top down)
  - Los tipos y propiedades de los framewoks

Bibliografía

- Clean Code: A Handbook of Agile Software Craftsmanship (Robert C. Martin Series)
- UML Distilled: A Brief Guide to the Standard Object Modeling Language
- Infrastructure as Code: Managing Servers in the Cloud
- Gray Literature:
  - https://alexmarket.medium.com/el-framework-sin-framework-294ee2888cb9
  - https://migabeta.wordpress.com/2017/01/23/que-son-los-frameworks/
  - https://www.seoestudios.es/blog/que-es-un-framework/
  - https://www.seoestudios.es/blog/que-es-un-framework/
  - https://astrolabit.com/el-precio-de-usar-frameworks
  - https://www.acens.com/wp-content/images/2014/03/frameworks-white-paper-acens-.pdf
  - https://programandoapps.com/tutorial/que-son-los-frameworks-de-desarrollo/
  - https://jordisan.net/blog/2006/que-es-un-framework/