Análisis de código de comportamiento

Luna Valdés García
Liliana Suárez Díaz
Noel Expósito Espina

La idea de analizar el código desde el lado de las personas

Objetivos

Priorizar deuda técnica

Favorecer la comunicación

Análisis sociales

¿De dónde sacamos los datos?

Fuentes

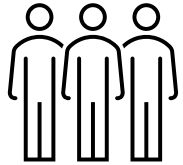
- Sistema de control de versiones

- Código

- Herramientas de gestion de producto
  - Jira
  - Azure DevOp

Diferencias respecto al análisis estático

- Impacto del mal código



- Nueva dimensión de estudio: las personas

Análisis de Hotspots

Qué deuda técnica representa el mayor riesgo

Dónde ocurren la mayor parte de pérdidas

Cuellos de botellas para desarrolladores

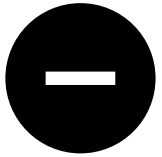
Medición de brechas de conocimiento

- ¿Refactorización o familiarización?
- ¿Código complejo o código con el que estamos poco familiarizados?

Calidad del código

- Depende del contexto.
- Las partes del código más cambiantes:
  - Delicadas
  - Mayor calidad
- Al solucionar un bug:
  - Calidad a segundo plano

Machine learning en BCA

- Beneficios 
  - Libera trabajo de revisar las PR, diferenciando PRs arriesgadas de las de bajo riesgo
  - Identifica patrones de implementación arriesgados
- Limitaciones 
  - Identificar La dirección de las dependencias (change coupling)

Integración de Behavioral Code Analysis

¿Cómo?

- Análisis por cada PR
- Forma de comunicación

¿Dónde?

- Sistemas Brown field
- Sistemas Green field

¿Por dónde empiezo?

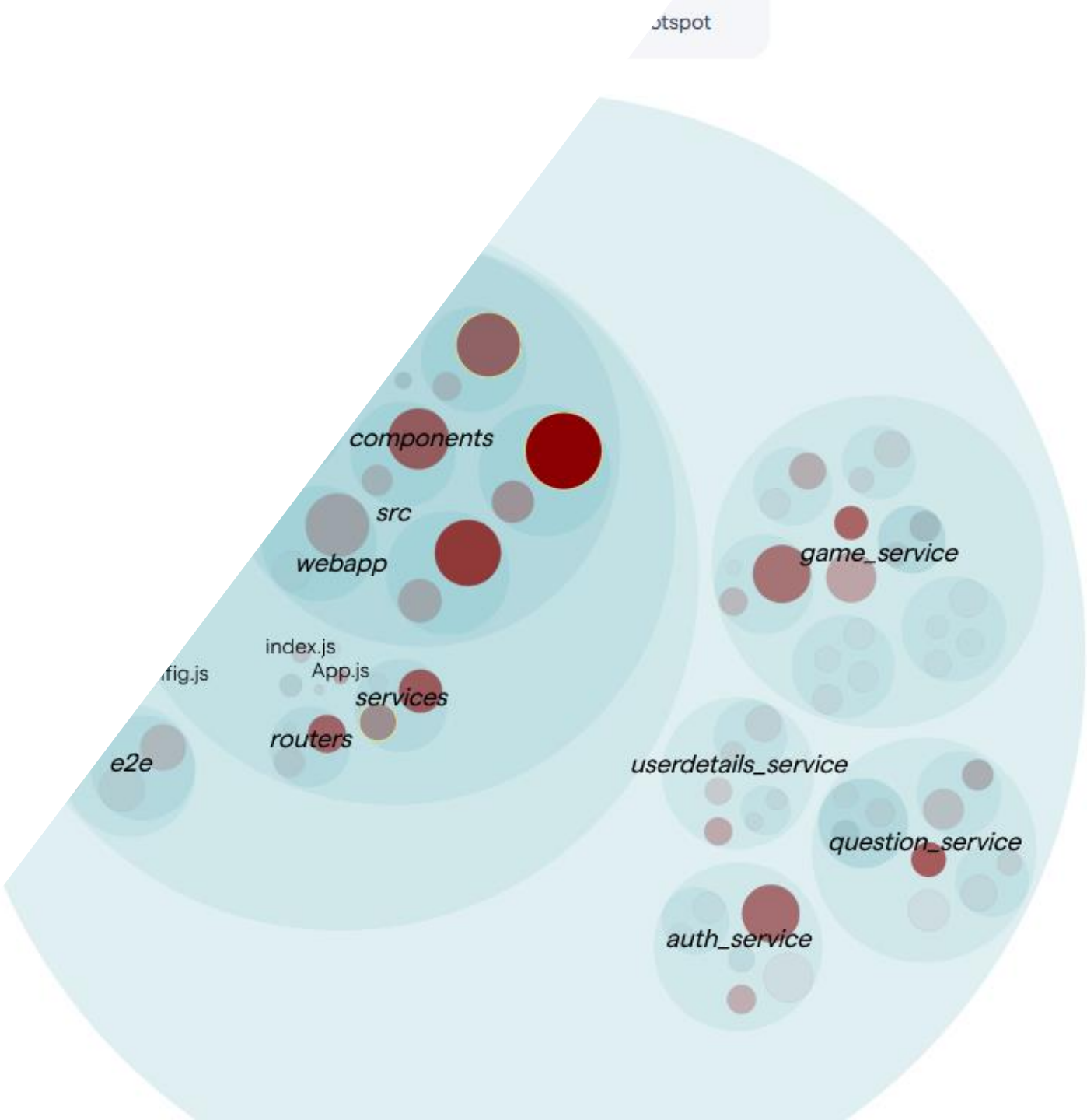
- Visualiza tu sistema
- Busca los hotspots
- Change coupling
- Ad hoc

Implicaciones éticas

- Se analizan equipos, no individuos
- Nunca para la clasificación de individuos
- Los análisis son para buscar mejoras, no para despedir personas

Code Maat

- Herramienta creada por Tornhill para hacer análisis
- Para cualquier VCS, mejor si es Git
- Usada en los análisis de CodeScene



Conclusión del gallu de Tornhill

Hay que asegurarse siempre de que optimizamos para entender

PREGUNTAS