

# **EVALUACIÓN DE HERRAMIENTAS DE DETECCIÓN Y CORRECCIÓN DE DEFECTOS DE DISEÑO**

**Alumnos:** Alberto Mendo Alonso  
Javier Sobrino Fernández

**Tutores:** Yania Crespo González-Carvajal  
Francisco Javier Pérez García





# ÍNDICE

1. Motivación
2. Objetivo
3. Introducción
  - a) Design smells
  - b) Taxonomía
4. Analisis de las herramientas
  - a) Target Artefact
  - b) Design Smells
  - c) Activity
5. Resumen de los análisis
6. Sitio Web
7. Conclusiones





# MOTIVACIÓN

- A medida que un sistema evoluciona, su estructura interna se degrada y surgen defectos de diseño (*Design smells*).
- Existen en el ámbito industrial y académico numerosas herramientas para detectar estos problemas. Debido a este gran número surge la necesidad de tener una clasificación de ellas.





# OBJETIVO

- El principal objetivo de este trabajo es obtener una guía de las principales herramientas de detección y corrección de defectos de diseño existentes, mediante su estudio y evaluación.
- Para mantener actualizada esta guía se crea un sitio web en el que se muestran los resultados.





# INTRODUCCIÓN

## A LOS DESIGN SMELLS

- *Design Smells*: soluciones pobres de diseño para problemas software generalizados y recurrentes.
- Pueden impedir su evolución, dificultando a los desarrolladores de software llevar a cabo cambios en el código.





# DESIGN SMELLS



- **Antipatterns**: Un antipatrón es una forma literaria que describe una solución recurrente que genera consecuencias negativas. (Brown et Al)
- **Disharmonies**: Defectos de diseño que afectan a entidades individuales tales como clases y métodos. (Lanza, Marinescu)
- **Bad Pattern Usage**: mala aplicación de un patrón en el código a analizar. (Joshua Kerievsky)
- **Bad Smells**: Síntomas de errores de diseño que indican la necesidad de refactorización. (Martin Fowler)
- **Metrics Warnings**: defectos de diseño localizados a través del cálculo de métricas.
- **Bad Coding Style**: Son violaciones en las convenciones de estilo o los estándares de código ya establecidos.

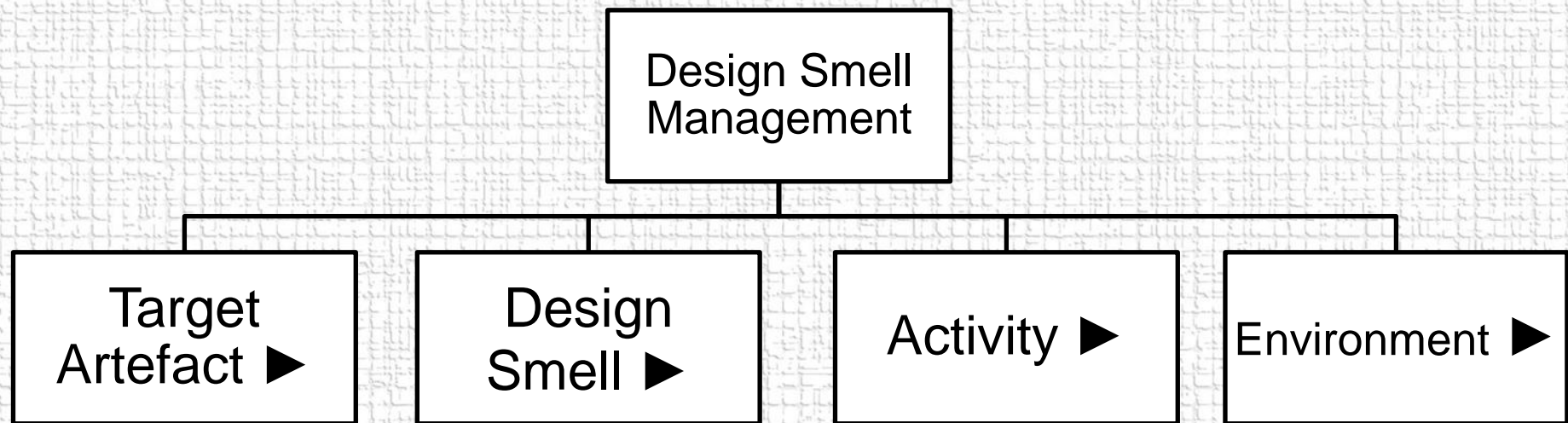


# TAXONOMÍA



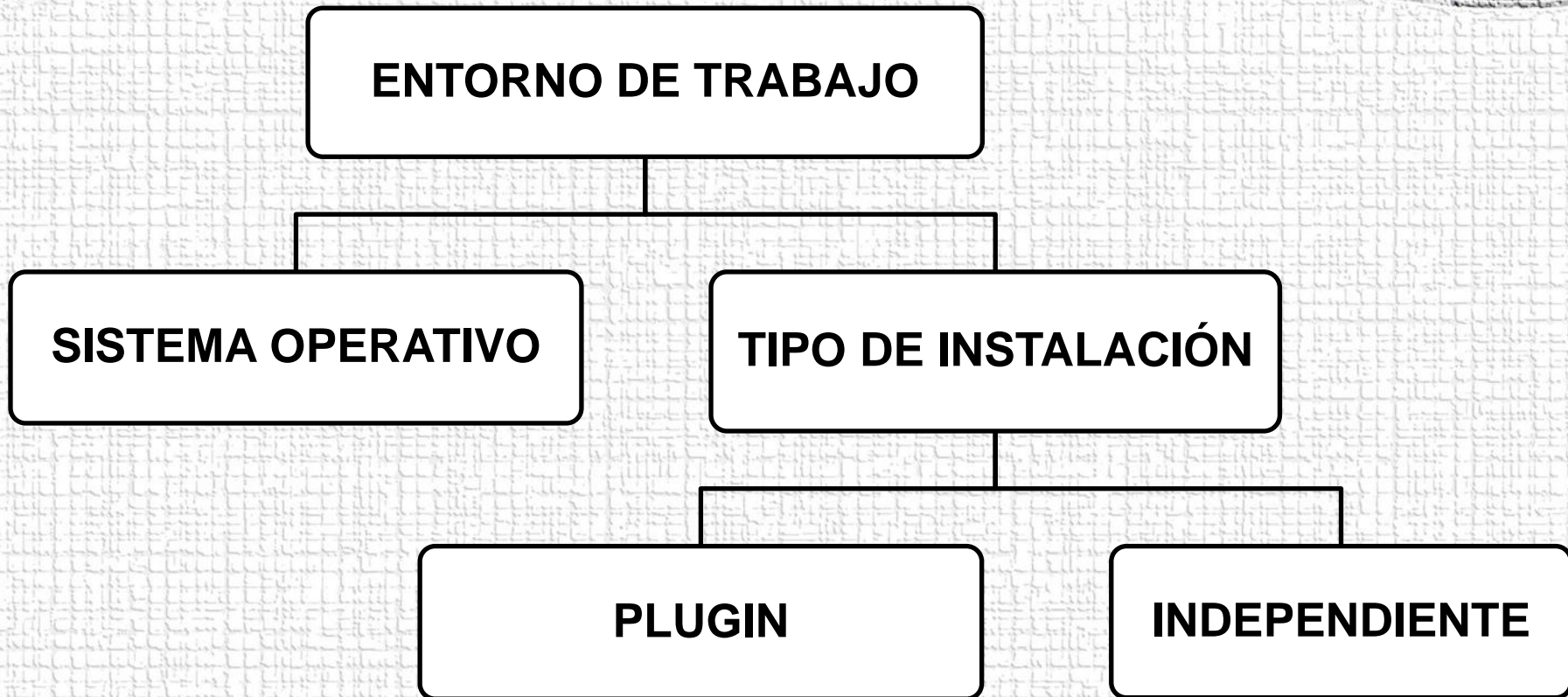


# TAXONOMÍA



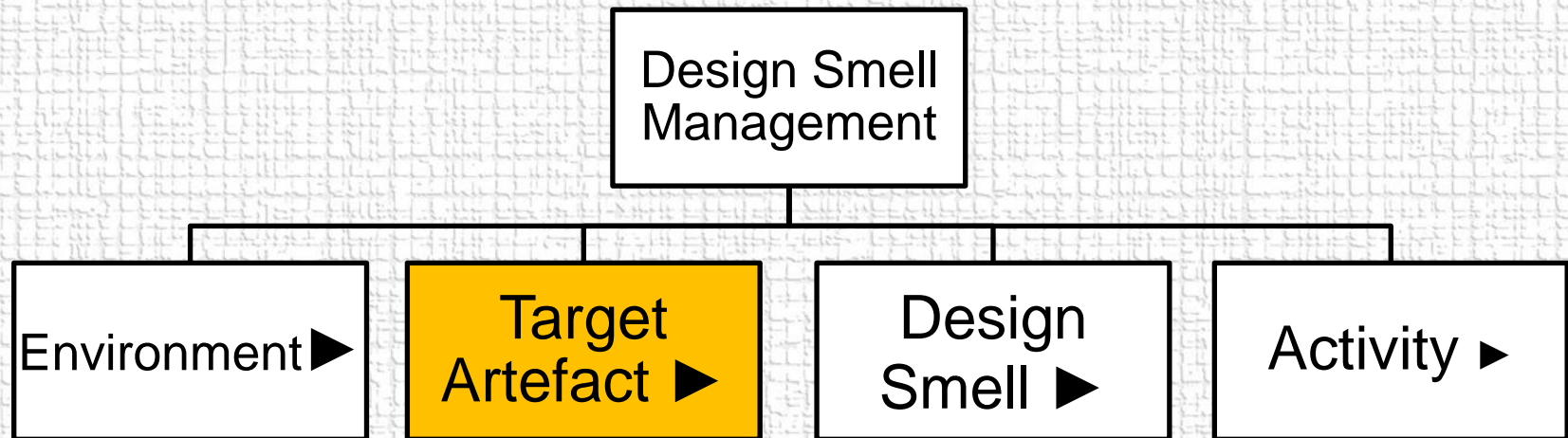


# TAXONOMÍA – ENTORNO DE TRABAJO





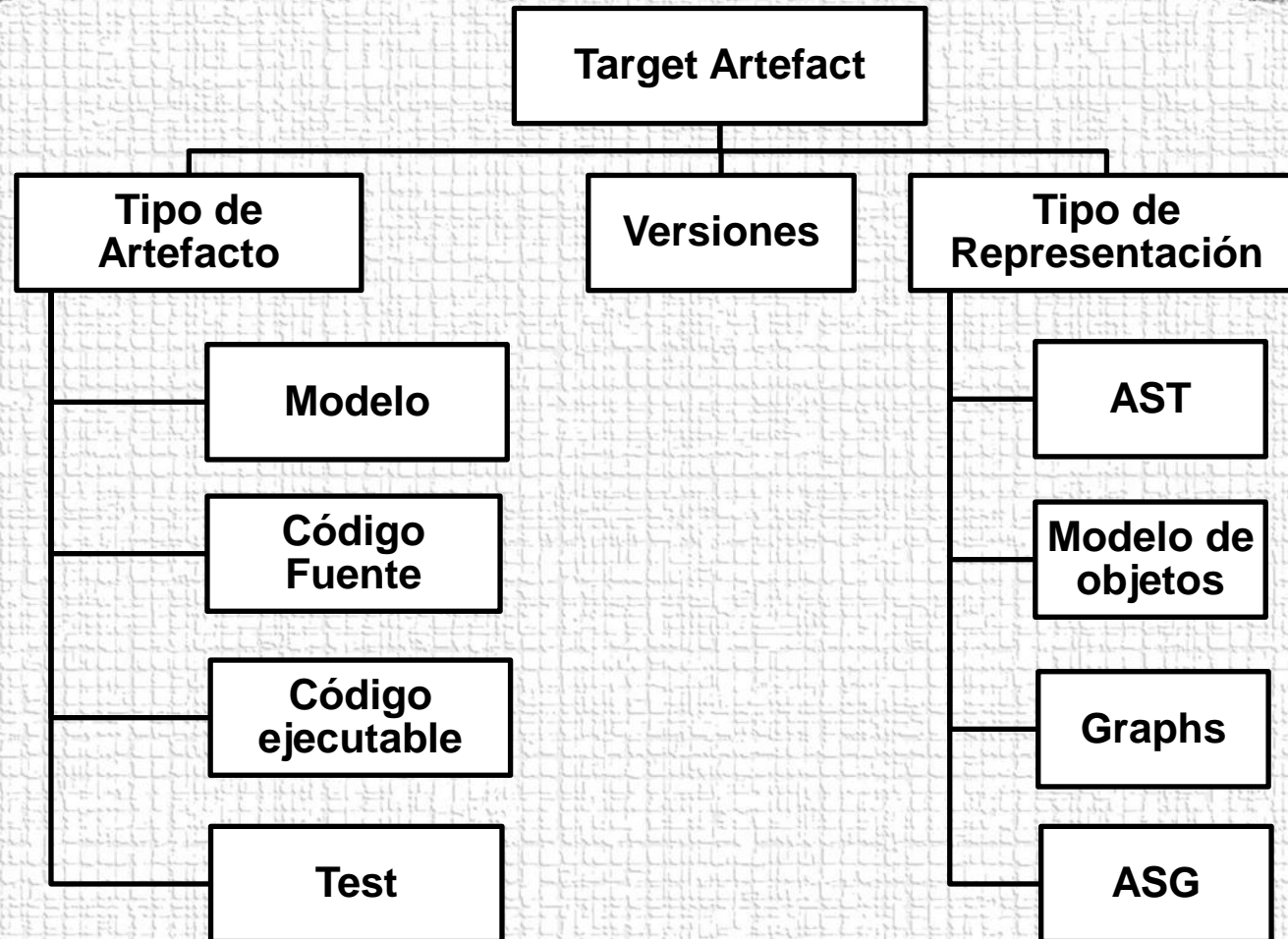
# TAXONOMÍA





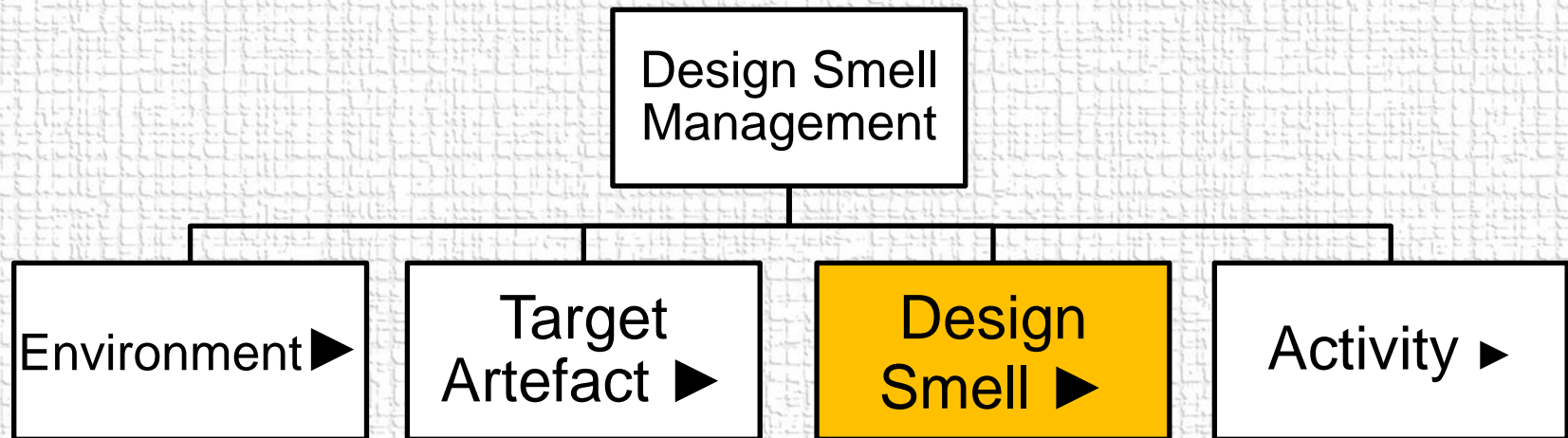


# TAXONOMÍA – TARGET ARTEFACT



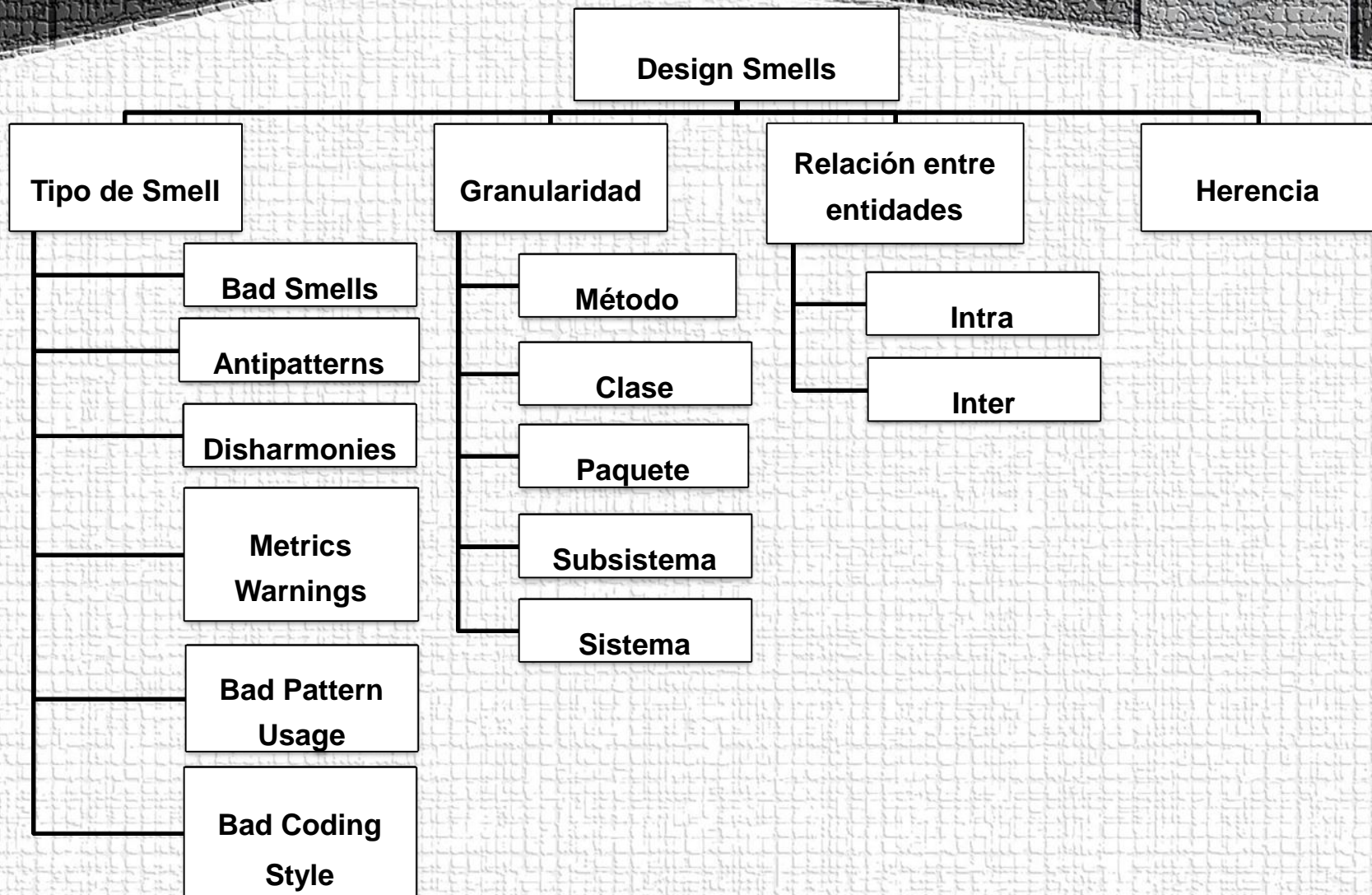


# TAXONOMÍA



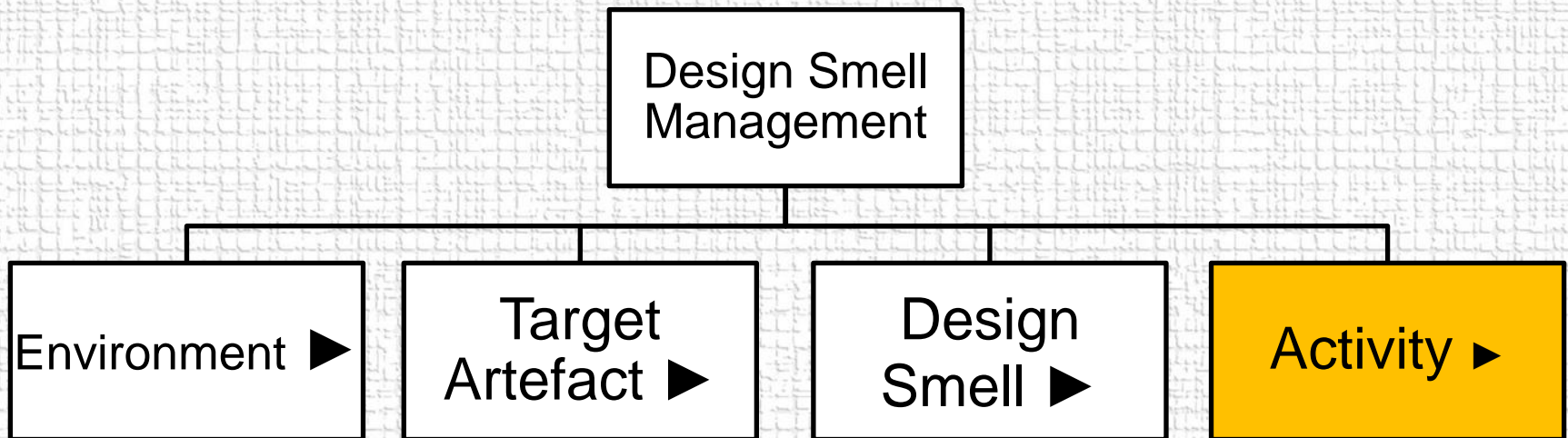


# TAXONOMÍA – DESIGN SMELLS





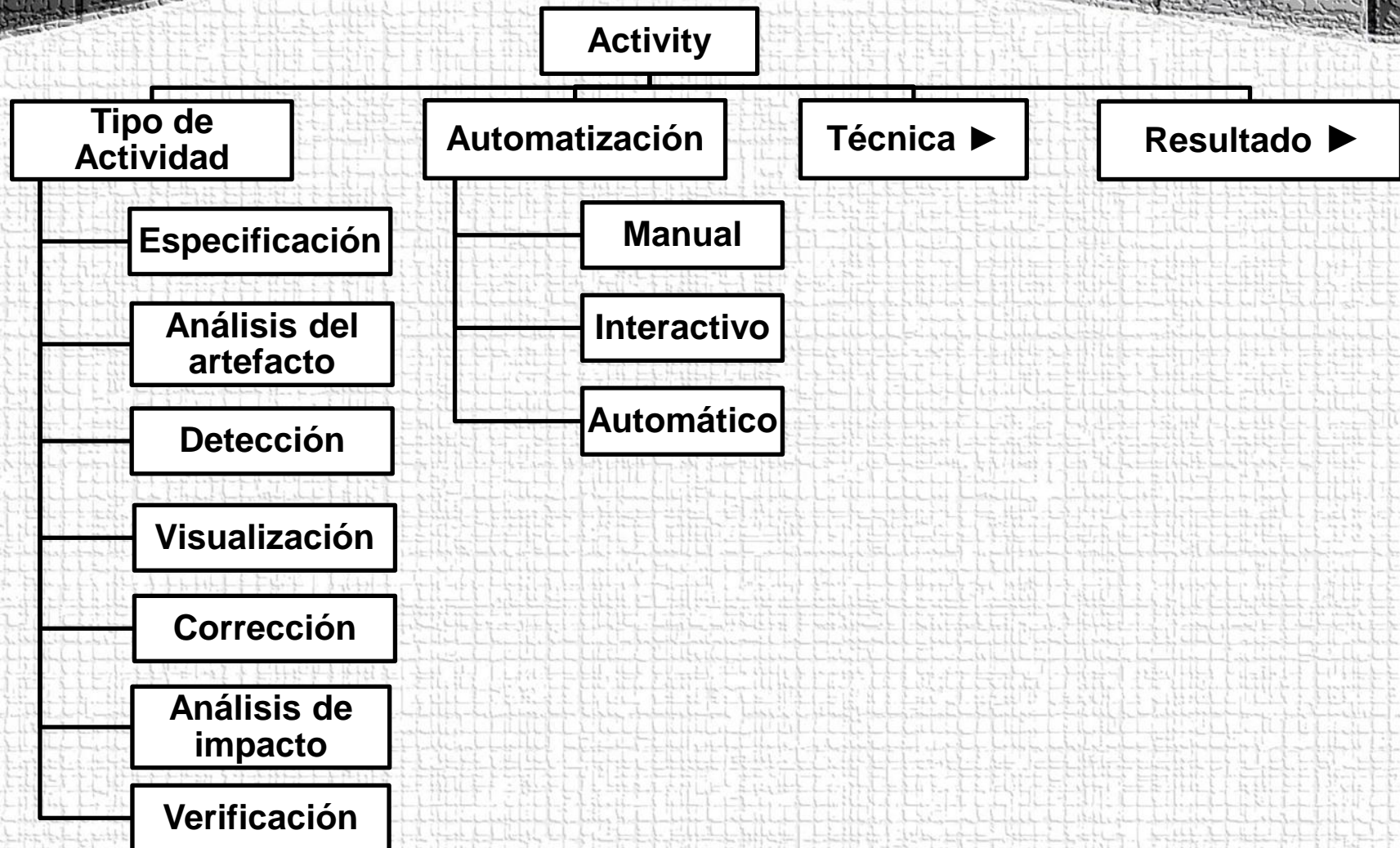
# TAXONOMÍA





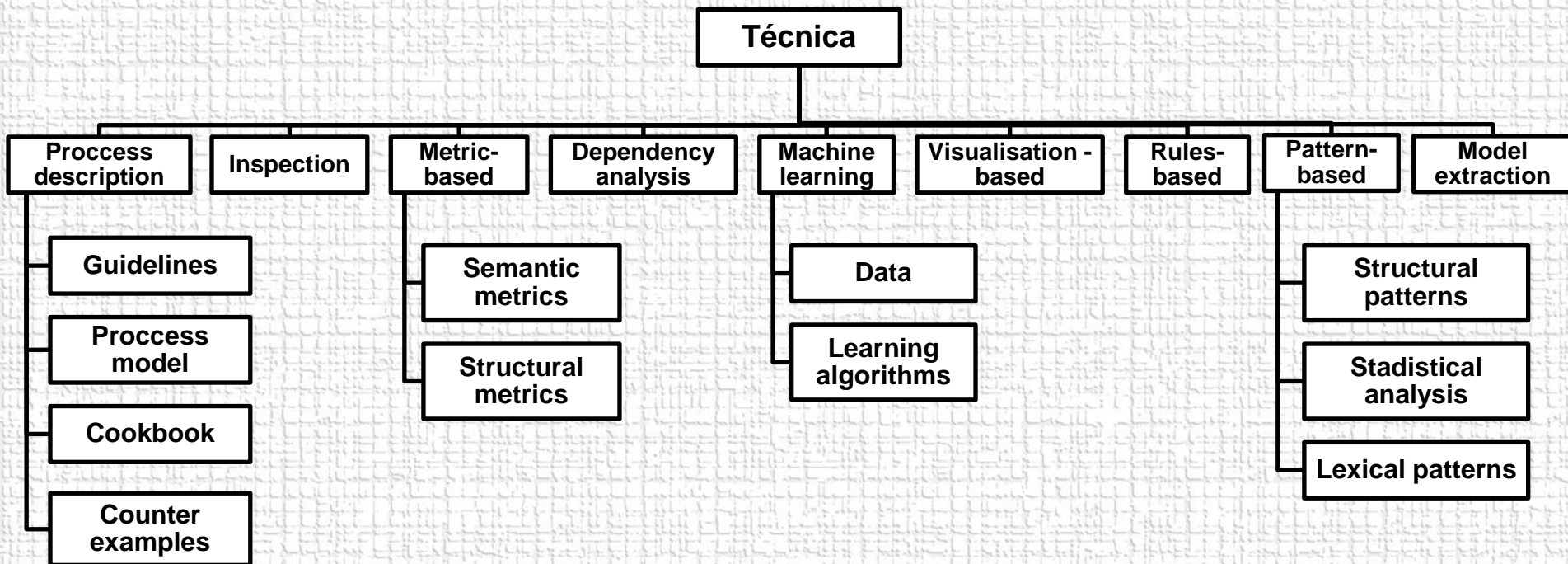


# TAXONOMÍA – ACTIVITY



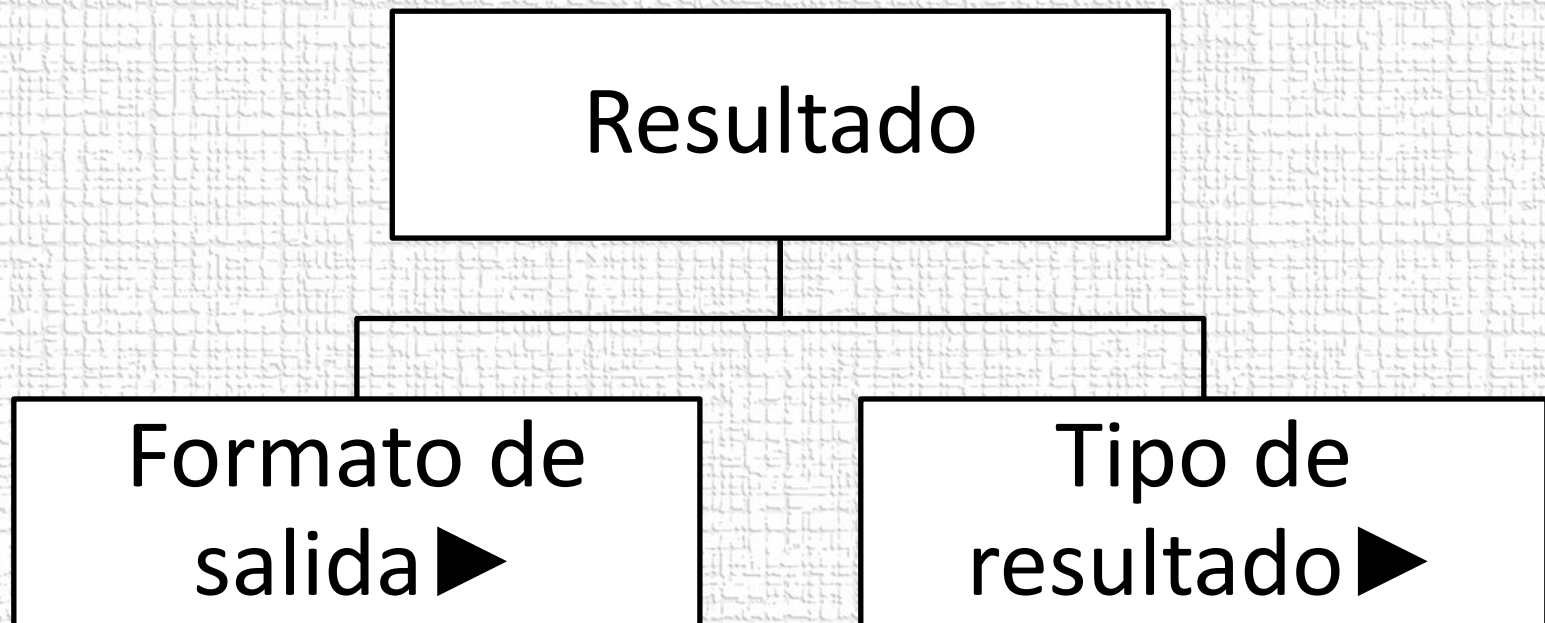


# TAXONOMÍA – ACTIVITY (TÉCNICA)



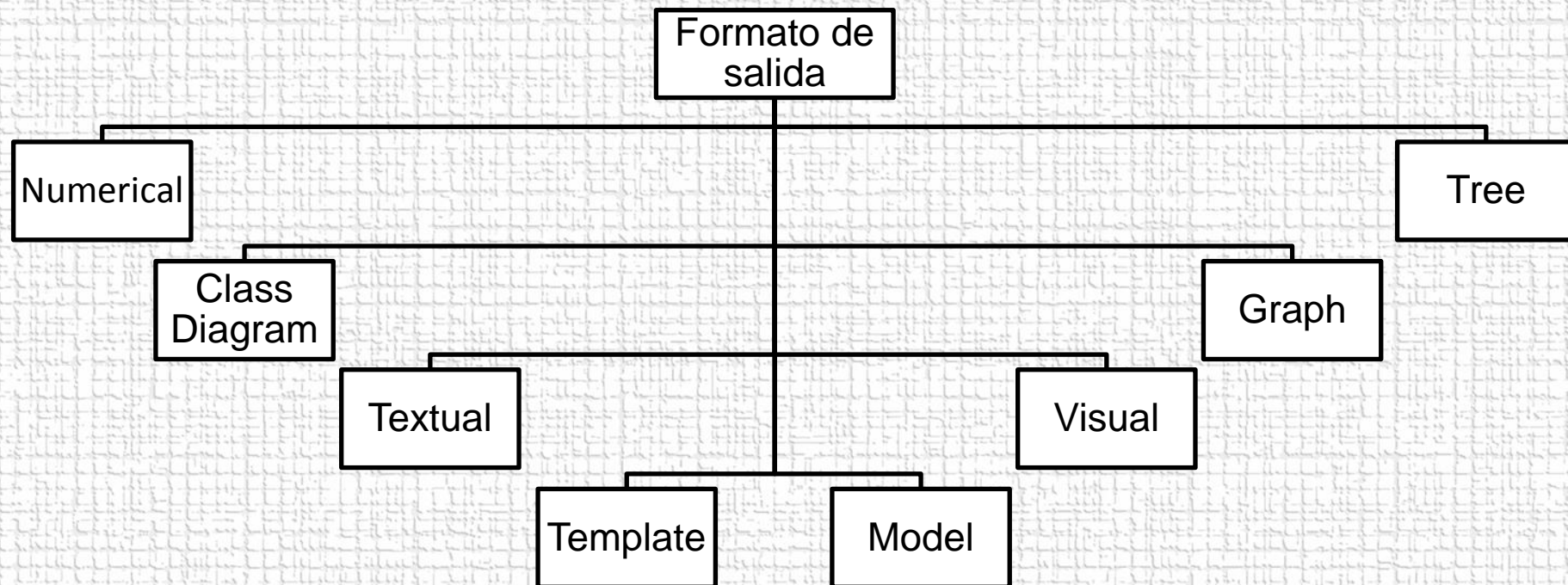


# TAXONOMÍA – ACTIVITY (RESULTADO)



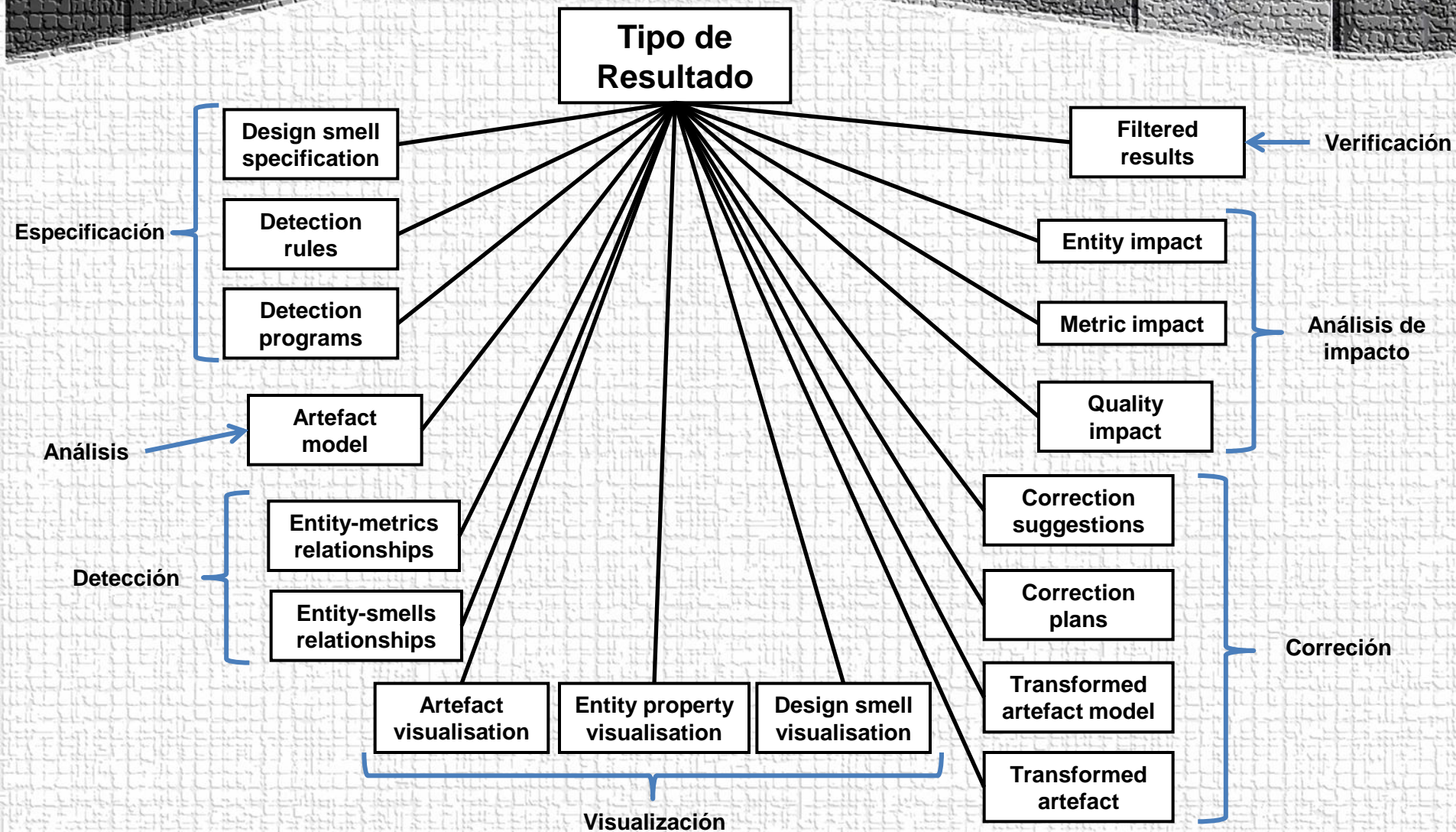


# TAXONOMÍA – ACTIVITY (RESULTADO)





# TAXONOMÍA – ACTIVITY (RESULTADO)





# ANÁLISIS DE HERRAMIENTAS





# ANÁLISIS DE LAS HERRAMIENTAS

**Propósito.**

**Información para el análisis.**

**Análisis.**

**c.1. Target Artefact.**

**c.2. Design Smells.**

**c.3. Activity.**

- Se ha llevado a cabo el análisis de 22 herramientas.
- Se va a detallar el análisis de 3 de ellas por ser de las más representativas.
- Al final se detallarán los resultados que se obtienen del estudio.





## ANÁLISIS DE HERRAMIENTAS (TARGET ARTEFACT)

HERRAMIENTA	ENTORNO DE TRABAJO	FORMATO DE ENTRADA	VERSIONES	TIPO DE REPRESENTACION
PMD	Independiente Plugin Eclipse Plugin jEdit Plugin NetBeans Otros IDE	Source Code (JAVA)  CPD admite: Java, JSP, C, C++, Fortran, Ruby y PHP code	NO	AST





## ANÁLISIS DE HERRAMIENTAS (DESIGN SMELL)

HERRAMIENTA	TIPO DE ERROR	GRANULARIDAD					RELACIÓN		HERENCIA
		Sistema	Subsistema	Paquetes	Clases	Métodos	Inter	Intra	
inCode	Bad Smells / Disharmonies Metrics Warning	SI	SI	SI	SI	SI	SI	SI	SI





# ANÁLISIS DE HERRAMIENTAS (ACTIVITY)

HERRAMIENTA	TIPO DE ACTIVIDAD	TÉCNICA	AUTOMATIZACIÓN	TIPO DE RESULTADO	FORMATO DE SALIDA
PMD	Especificación	Process description	Interactive	Detection Rules	Textual
			Manual	Detection Programs	Textual
	Análisis	Model Extraction	Fully-automated	-	-
	Detección	Rules-Based	Interactive	Entity-smells relationships	Textual / Visual





# ANÁLISIS DE HERRAMIENTAS (ACTIVITY)

HERRAMIENTA	TIPO DE ACTIVIDAD	TÉCNICA	AUTOMATIZACIÓN	TIPO DE RESULTADO	FORMATO DE SALIDA
jDeodorant	Detección	Metrics-Based	Interactive	Entity-smells relationships	Textual, numerical
	Corrección	Refactoring Suggestions	Interactive	Refactoring actions	Textual, Model
	Análisis de impacto	Metrics-Based	Interactive	Metric-Impact	Numerical





# RESULTADOS DEL ESTUDIO





# TABLA COMPLETA

## CARACTERÍSTICA TARGET ARTEFACT

HERRAMIENTA	ENTORNO DE TRABAJO	FORMATO DE ENTRADA	VERS.	TIPO DE REPRESENTACION
Analyst4j	Plugin Eclipse/Independiente	Source Code (Java)	NO	-
ArgoUML	Independiente	Source Code (JAVA, C, C++) Model (UML, Zargo, XMI, XML)	NO	-
Argus CodeWatch	Plugin Eclipse	Source Code (Java)	NO	-
CodePro Analytix	Plugin Eclipse	Source Code (Java)	NO	-
Cultivate	Plugin Eclipse	Source Code (Java)	NO	AST
Eclipse	Independiente	Source Code (Java, C, C++,Python)	SI	AST
FxCop	Independiente / Plugin Visual Studio	Microsoft .NET Assemblies	NO	Object Model, Graphs
inCode	Plugin Eclipse	Source Code (Java)	NO	-
jDeodorant	Plugin Eclipse	Source Code (Java)	NO	AST
JRefactory	Independiente / Plugin jEdit / Plugin NetBeans 3.6. / Plugin jBuilderX	Source Code (Java) Executable Code	NO	AST





# TABLA COMPLETA

## CARACTERÍSTICA TARGET ARTEFACT

HERRAMIENTA	ENTORNO DE TRABAJO	FORMATO DE ENTRADA	VERS.	TIPO DE REPRESENTACION
M2 Resource Standard Metrics	Independiente Plugin de: Eclipse/ Visual Studio 6/ Visual Studio .NET	Source Code (C,C++,C#, Java)	NO	Object Model
PMD	Independiente / Plugin Eclipse / Plugin jEdit / Plugin NetBeans / Otros IDE	Source Code (Java) (CPD admite Java, JSP, C, C++, Fortran, Ruby y PHP code)	NO	AST
Reek	Independiente (Linux)	Source Code (Ruby)	NO	AST
RevJava	Independiente	Executable Code	NO	AST
Roodi	Independiente	Source Code (Ruby)	NO	AST
STAN	Independiente / Plugin Eclipse	Executable Code	NO	-
SA4J	Independiente	Source Code (Java (.class)) Executable Code	NO	-
Structure101	Independiente / Plugin Eclipse / Plugin IntelliJ / Aplicación web	Source Code Executable Code	SI	-
StyleCop	Plugin Visual Studio / MSBuild	Source Code (C#)	NO	-
Together	Plugin Eclipse	UML Model Source Code (Java/C/C++/CORBA)	SI	Object Model
TRex	Independiente/Plugin Eclipse	Source Code (TTCN-3)	NO	AST
XRefactory	Plugin jEdit / Emacs / Xemacs	Source Code (JAVA/C/C++)	NO	-





# RESULTADOS DEL TRABAJO:

## CARACTERÍSTICA TARGET ARTEFACT

- Gran número de las herramientas han sido desarrolladas como plugin de eclipse.
- Principalmente analizan código java. Destacan Reek y Roodi por analizar código Ruby y TRex por analizar código TTCN-3.
- Sólo tres herramientas permiten control de versiones.
- Escasa información acerca de la representación interna del artefacto de entrada.
- AST es la representación mayoritaria de las herramientas de las que se tiene información.





# RESULTADOS DEL TRABAJO: CARACTERÍSTICA DESIGN SMELLS

- Predomina la búsqueda de Bad Coding Style y Metric Warnings.
- Together y Cultivate son dos de las completas.
- Argus CodeWatch y jDeodorant son de las más simples.
- La granularidad se concentra principalmente en los niveles de método y clase.





# RESULTADOS DEL TRABAJO:

## CARACTERÍSTICA ACTIVITY

- Todas incorporan la actividad de Detección. M2 RSM y Xrefactory sólo incorporan esta actividad.
- Un gran número de las herramientas incorporan la actividad de Especificación.
- jDeodorant y SA4j destacan por añadir Análisis de Impacto.





# SITIO WEB





# SITIO WEB - OBJETIVOS

- Mostrar la información recopilada acerca de las herramientas de análisis de código.





# SITIO WEB - CARACTERÍSTICAS

## Técnicas:

- Se ha creado mediante el Gestor de contenidos Joomla!
- Se utiliza una base de datos MySQL.
- Contiene un buscador interno y control de usuarios.

## Funcionalidad:

- Permite actualizar los datos de las herramientas.
- Permite agregar nuevos análisis de herramientas.
- Permite realizar comentarios sobre los análisis existentes.





# DESIGN SMELL MANAGEMENT TOOLS

[Inicio](#) [Estudio Previo](#) [Design Smells](#) [Analizadas](#) [Otras Herramientas](#)

## ACCESO DE USUARIO

[Acceder](#)[Recordar contraseña](#)[Recordar usuario](#)[Crea una cuenta](#)

## MENÚ PRINCIPAL

- ▶ [Inicio](#)
- ▶ [Estudio Previo](#)
- ▶ [Design Smells](#)
- ▶ [Analizadas](#)
- ▶ [Otras Herramientas](#)

## DESIGN SMELL MANAGEMENT TOOLS



A medida que un sistema evoluciona y sufre modificaciones, su estructura interna, su diseño, se degrada, empeorando a su vez la mantenibilidad y la reusabilidad del mismo. Existen en el ámbito industrial y académico numerosas herramientas para detectar estos problemas, conocidos como defectos de diseño.

En esta web se pretende mostrar el estudio y evaluación de una serie de herramientas de detección de defectos de diseño realizado inicialmente por Alberto Mendo y Javier Sobrino, dos alumnos de la Escuela Universitaria de Ingeniería Informática perteneciente a la Universidad de Valladolid. Para dicho análisis se parte de un estudio previo, ya existente, en el que se detallan cuáles son las características deseables y comunes para una herramienta de este tipo y, por lo tanto, las características evaluables, y en el que ya se ha realizado una evaluación de un pequeño número de herramientas. Este estudio ha sido realizado por Javier Pérez, Naouel Moha, Tom Mens y Carlos López y se encuentra en el documento *A classification framework and survey for design smell management*. En el apartado **Estudio Previo** de esta web se pueden encontrar fragmentos de dicho artículo que nos ayudarán a entender los análisis aquí mostrados y sus respectivas tablas.

Este estudio no está cerrado. Es por ello que ponemos a disposición de los usuarios registrados un formulario mediante el cuál podrás solicitarnos permisos para la publicación de un nuevo artículo, así como la modificación de alguno de los ya existentes.

La web se compone, además del apartado **Estudio Previo** ya mencionado, de otros apartados que son **Analizadas**, en el que se incluyen los análisis y tablas de las herramientas listados, **Otras herramientas**, en el que se pueden encontrar una serie de herramientas que inicialmente se pretendían analizar y que por diversos motivos han sido descartadas, y el apartado **Design Smells**, en el que se pueden encontrar los listados que se han utilizado para clasificar los tipos de errores que las herramientas detectan como son *Antipatterns*, *Bad Smells* y *Disharmonies*.

[Arriba](#)

 Universidad de Valladolid |  Escuela Técnica Superior de Ingeniería Informática | [Mapa del sitio](#)

Contactar con nosotros: [Grupo GIRO](#) | [Javier Pérez](#) | [Yania Crespo](#)

[designsmellstools.co.cc](http://designsmellstools.co.cc) | Diseñado y realizado por [Alberto Mendo](#) y [Javier Sobrino](#)



Universidad de Valladolid



# CONCLUSIONES FINALES





# CONCLUSIONES

- La comprensión del trabajo dado inicialmente ha sido una de las partes más complejas por la aparición de un gran numero de conceptos nuevos para nosotros.
- Se desconocía la existencia de este tipo de herramientas.
- La búsqueda inicial de herramientas ha resultado más sencilla de lo esperado.
- Existen herramientas de difícil instalación como Reek y Roodi, y de difícil manejo como Xrefactory.
- El apartado más complicado de completar durante el análisis ha sido el tipo de representación por la falta de información disponible al respecto.
- El gran número de herramientas ha provocado que el proyecto se alargara más de lo esperado.
- Se ha creado una guía útil para poder elegir la herramienta adecuada a las necesidades de cada desarrollador.





# **EVALUACIÓN DE HERRAMIENTAS DE DETECCIÓN Y CORRECCIÓN DE DEFECTOS DE DISEÑO**

**Alumnos:** Alberto Mendo Alonso  
Javier Sobrino Fernández

**Tutores:** Yania Crespo González-Carvajal  
Francisco Javier Pérez García

