30th IEEE/ACM International Conference on Program Comprehension (ICPC 2022)

Code Smells in Elixir: Early Results from a Grey Literature Review

Lucas Francisco da Matta Vegi
Marco Tulio Valente

Federal University of Minas Gerais (UFMG)
Belo Horizonte, Minas Gerais, Brazil

aSERG
APPLIED SOFTWARE ENGINEERING RESEARCH GROUP
/ UFMG

May 17th, 2022

What is Elixir?

- Elixir is a **modern functional programming language**:
  - high performance in parallel and distributed environments
  - mix of inspirations

What is Elixir?

- The **popularity of functional languages is on the rise** in the industry

- More than **600 companies** already use Elixir in **production code**

Problem

- It is natural to expect that Elixir developers make **bad design decisions**
  - Code smells;
  - Synonyms: bad smells, anti-patterns, and code anomalies.

- There are no papers in the literature focused on studying the **internal quality of Elixir systems**
  - few works on software quality to functional language

Research questions (RQs)

- **RQ1:** Do Elixir developers discuss traditional code smells?
- **RQ2:** Do Elixir developers discuss other smells?
- **RQ3:** Does a well-known static code analysis tool for Elixir detect code smells?

Methodology (RQ1 & RQ2)

```
("Elixir") AND
("code smell" OR "code smells" OR "bad smell" OR "bad
     smells" OR "anti-pattern" OR "anti-patterns" OR
     "antipattern" OR "antipatterns" OR "bad-practice" OR
     "bad-practices" OR "bad practice" OR "bad practices")
```

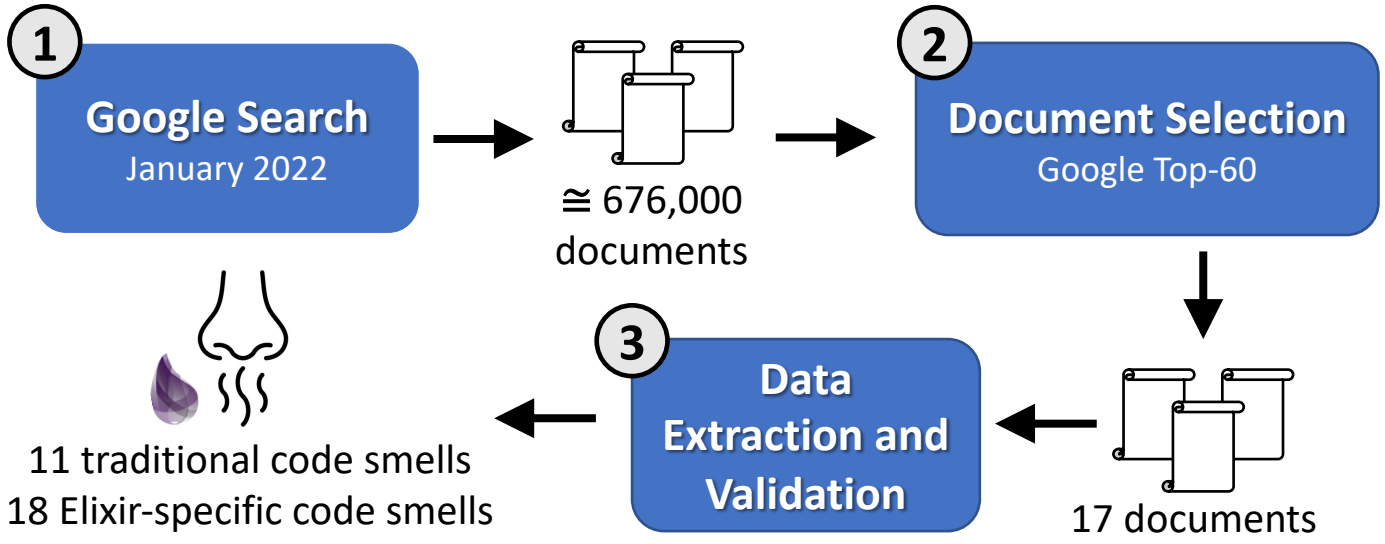


Figure 1: Overview of the grey literature methods

Methodology (RQ3)

- We carefully analyzed **Credo documentation** (version 1.6.2)
  - **Aiming:** find warnings for code smells

https://hexdocs.pm/credo

RQ1: Do Elixir developers discuss traditional code smells?

Table 1: Traditional code smells (grey literature)

Code Smell in Elixir	Documents	#
Comments	D1, D10, D12, D14	4
Long Parameter List	D1, D16	2
Feature Envy	D1, D6	2
Shotgun Surgery	D1, D17	2
Duplicated Code	D1	1
Long Function	D1	1
Large Class	D1	1
Inappropriate Intimacy	D1	1
Divergent Change	D1	1
Speculative Generalization	D1	1
Primitive Obsession	D3	1

Eleven (out of 22) smells are discussed

RQ1: Do Elixir developers discuss traditional code smells?

Table 1: Traditional code smells (grey literature)

Code Smell in Elixir	Documents	#
Comments	D1, D10, D12, D14	4
Long Parameter List	D1, D16	2
Divergent Change	D1	1
Speculative Generalization	D1	1
Primitive Obsession	D3	1

Eleven (out of 22) smells are discussed

"A long parameters list is one of many potential bad smells [...]. In object-oriented languages like Ruby or Java, we could easily define classes that help us solve this problem. Elixir does not have classes but because it is easy to extend, we can define our own types."

RQ2: Do Elixir developers discuss other smells?

Table 2: Elixir-specific code smells (grey literature)

	Smell	Description	Docs
Design-related smells	GenServer Envy	Using a Task or Agent but handling them like GenServers	D8
	Agent Obsession	When the responsibility for interacting with an Agent process is spread across the system	D8
	Unsupervised process	Library that creates process outside a supervision tree, not allowing users to fully control their apps	D5
	Large messages between processes	Processes that exchange long messages frequently	D13
	Complex multi-clause function	Function with many guard clauses and pattern matching	D10
	Complex API error handling	Function that handles a large number of error types returned by an API endpoint, making it complex	D2
	Exceptions for control-flow	A library that forces clients to handle control-flow exceptions	D5, D11
	Untested polymorphic behavior	Function with a generic Protocol type parameter, but that does not have guards verifying its behavior	D7
	Code organization by process	Library unnecessarily organized as a process, instead as modules and functions	D5
	Data manipulation by migration	Low cohesive module that performs both data and structural changes in a DB schema via Ecto.Migration	D9
Low-level concerns smells	Working with invalid data	Function that does not validate its parameters and therefore can produce non-predicted behaviors	D5
	Map/struct dynamic access	Function that dynamically accesses values from a non-existent field in a struct or map	D7
	Unplanned value extraction	Function that does not force a crash when an incorrect value is extracted from a URL query string	D7
	Modules with identical names	Modules with conflicting names, preventing their simultaneous load	D5
	Unnecessary macro	Using macros instead of functions and structs	D5
	App configuration for code libs	Function that uses global configuration mechanisms, preventing clients from reusing it flexibly	D5
	Compile-time app configuration	Function that uses module attributes to configure itself, therefore preventing run-time configurations	D5
	Dependency with "use" when an "import" is enough	Module that requires knowledge of its internal dependencies to understand its behavior	D5

10 smells

8 smells

For more details: https://github.com/lucasvegi/Elixir-Code-Smells

RQ3: Does a well-known static code analysis tool for Elixir detect code smells?

- Credo is able to identify **only three (out of 40) smells**

- **Examples**:
  - Duplicated code and Long Parameter List **(traditional)**;
  - Compile-time app configuration **(low-level concern)**.

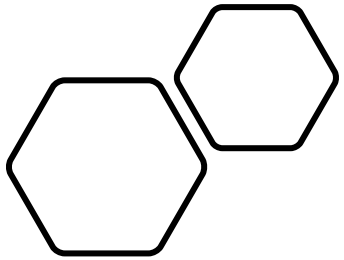
Future work

1) We plan to **extend and validate our catalog of smells:**
   - Mining Elixir OSS (issues, pull requests and commits)
   - Surveys and interviews with Elixir developers

2) **Extend Credo** tool for detecting smells in Elixir

3) Investigate **code smells in other modern functional languages**

Thank you!



Lucas Francisco da Matta Vegi

(lucasvegi@dcc.ufmg.br)

preprint: https://arxiv.org/abs/2203.08877

aSERG APPLIED SOFTWARE ENGINEERING RESEARCH GROUP / UFMG