Towards a Taxonomy of Grammar Smells

V. Zaytsev @ SLE’17 @ SPLASH
joint work with M. Stijlaart
Mäntylä-Vanhanen-Lassenius Taxonomy of Bad Smells

- Bloaters
- OO abusers
- Change preventers
- Dispensables
- Encapsulators
- Couplers
- Others
  - incomplete + comments
Our Taxonomy

- Organisation
  - Convention
  - Notation
  - Parsing
  - Duplication

- Navigation
  - Spaghetti
  - Shortage
  - Mixture

- Structure
  - Proxy
  - Dependency
  - Complexity
**Grammar Smells**

<table>
<thead>
<tr>
<th>Misnaming</th>
<th>Underuse</th>
<th>Overuse</th>
<th>Singleton</th>
<th>Chant</th>
<th>Almost</th>
<th>Uncluster</th>
<th>Neatness</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>stmt ::= IF condition THEN b;</code></td>
<td><code>foo = one of bar</code></td>
<td><code>bar</code></td>
<td><code>\{a\}</code></td>
<td><code>\{b\}</code></td>
<td><code>\{c\}</code></td>
<td><code>\{d\}</code></td>
<td></td>
</tr>
<tr>
<td>`a ::= b</td>
<td>c;`</td>
<td><code>\{e\}</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>s ::= d;</code></td>
<td><code>Exp</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Factoring**

| `st1 ::= IF c THEN s`           | `Exp`                            |          |          |        |        |           |
| `st2 ::= IF c THEN s`           | `Exp`                            |          |          |        |        |           |
| `st ::= [s1,s2]`                | `Exp`                            |          |          |        |        |           |
| `a ::= b | c;`                        | `\{e\}`                         |          |          |        |        |           |
| `s ::= d;`                       | `Exp`                            |          |          |        |        |           |

**ZigZag**

| `a ::= b | c;`                        | `\{e\}`                         |          |          |        |        |           |
| `s ::= d;`                       | `Exp`                            |          |          |        |        |           |

**Ouroboros**

| `name := id | stdname;`                   | `Diamond`                        |          |          |        |        |           |
| `r ::= l | r;`                      | `\{e\}`                         |          |          |        |        |           |
| `l ::= b | b;`                       | `\{e\}`                         |          |          |        |        |           |

**Grammar**

| `st ::= IF condition THEN b;`   | `foo = one of bar`              | `bar`   | `\{a\}`  | `\{b\}`| `\{c\}` | `\{d\}`   |
| `a ::= b | c;`                        | `\{e\}`                         |          |          |        |        |           |
| `s ::= d;`                       | `Exp`                            |          |          |        |        |           |

**Uncluster**

| `st ::= IF condition THEN b;`   | `foo = one of bar`              | `bar`   | `\{a\}`  | `\{b\}`| `\{c\}` | `\{d\}`   |
| `a ::= b | c;`                        | `\{e\}`                         |          |          |        |        |           |
| `s ::= d;`                       | `Exp`                            |          |          |        |        |           |

**Grammar**

| `st ::= IF condition THEN b;`   | `foo = one of bar`              | `bar`   | `\{a\}`  | `\{b\}`| `\{c\}` | `\{d\}`   |
| `a ::= b | c;`                        | `\{e\}`                         |          |          |        |        |           |
| `s ::= d;`                       | `Exp`                            |          |          |        |        |           |

**Uncluster**

| `st ::= IF condition THEN b;`   | `foo = one of bar`              | `bar`   | `\{a\}`  | `\{b\}`| `\{c\}` | `\{d\}`   |
| `a ::= b | c;`                        | `\{e\}`                         |          |          |        |        |           |
| `s ::= d;`                       | `Exp`                            |          |          |        |        |           |

**MiddleStart**

| `st ::= IF condition THEN b;`   | `foo = one of bar`              | `bar`   | `\{a\}`  | `\{b\}`| `\{c\}` | `\{d\}`   |
| `a ::= b | c;`                        | `\{e\}`                         |          |          |        |        |           |
| `s ::= d;`                       | `Exp`                            |          |          |        |        |           |

**MiddleStart**

| `st ::= IF condition THEN b;`   | `foo = one of bar`              | `bar`   | `\{a\}`  | `\{b\}`| `\{c\}` | `\{d\}`   |
| `a ::= b | c;`                        | `\{e\}`                         |          |          |        |        |           |
| `s ::= d;`                       | `Exp`                            |          |          |        |        |           |

**MiddleStart**

| `st ::= IF condition THEN b;`   | `foo = one of bar`              | `bar`   | `\{a\}`  | `\{b\}`| `\{c\}` | `\{d\}`   |
| `a ::= b | c;`                        | `\{e\}`                         |          |          |        |        |           |
| `s ::= d;`                       | `Exp`                            |          |          |        |        |           |

**MiddleStart**

| `st ::= IF condition THEN b;`   | `foo = one of bar`              | `bar`   | `\{a\}`  | `\{b\}`| `\{c\}` | `\{d\}`   |
| `a ::= b | c;`                        | `\{e\}`                         |          |          |        |        |           |
| `s ::= d;`                       | `Exp`                            |          |          |        |        |           |

**MiddleStart**

| `st ::= IF condition THEN b;`   | `foo = one of bar`              | `bar`   | `\{a\}`  | `\{b\}`| `\{c\}` | `\{d\}`   |
| `a ::= b | c;`                        | `\{e\}`                         |          |          |        |        |           |
| `s ::= d;`                       | `Exp`                            |          |          |        |        |           |

**MiddleStart**

| `st ::= IF condition THEN b;`   | `foo = one of bar`              | `bar`   | `\{a\}`  | `\{b\}`| `\{c\}` | `\{d\}`   |
| `a ::= b | c;`                        | `\{e\}`                         |          |          |        |        |           |
| `s ::= d;`                       | `Exp`                            |          |          |        |        |           |

**MiddleStart**

| `st ::= IF condition THEN b;`   | `foo = one of bar`              | `bar`   | `\{a\}`  | `\{b\}`| `\{c\}` | `\{d\}`   |
| `a ::= b | c;`                        | `\{e\}`                         |          |          |        |        |           |
| `s ::= d;`                       | `Exp`                            |          |          |        |        |           |

**MiddleStart**

| `st ::= IF condition THEN b;`   | `foo = one of bar`              | `bar`   | `\{a\}`  | `\{b\}`| `\{c\}` | `\{d\}`   |
| `a ::= b | c;`                        | `\{e\}`                         |          |          |        |        |           |
| `s ::= d;`                       | `Exp`                            |          |          |        |        |           |

**MiddleStart**

| `st ::= IF condition THEN b;`   | `foo = one of bar`              | `bar`   | `\{a\}`  | `\{b\}`| `\{c\}` | `\{d\}`   |
| `a ::= b | c;`                        | `\{e\}`                         |          |          |        |        |           |
| `s ::= d;`                       | `Exp`                            |          |          |        |        |           |

Interactive at [http://blebok.github.io/grass](http://blebok.github.io/grass)

Mats Stijlart & Vadim Zaytsev
http://slebok.github.io/grass
- **Misformat**
  - misleading formatting

- **Misnomer**
  - problems with naming policies

- **SayMyName**
  - a misspelling or improper use of the natural language

- **ZigZag**
  - mixing horizontal and vertical styles

- **Splat**
  - nonterminal fragments scattered over the grammar
abc:

def;

ghi:

Jkl;

mno:

pqr;
/ Organisation / Notation

- Underuse
- Overspec
- Priorities
- Singleton
- Combo
- Chant
- Deprecated
- Exotic
/ Organisation / Parsing

- Factoring
- 1SidedRecursion
- Superset
- Shotgun
- NoDefault
- Action
  - semantic action is code
define expression
    [expression] [op] [expression]
|   [id] [expression+]
|   if [expression] then [expression] else [expression]
|   ( [expression] )
|   [id]
|   [number]
end define
/ Organisation / Duplication

- **Echo**
  - same rule occurs twice or more
- **Clone**
  - same right hand side occurs twice or more
- **Foldable**
  - nonterminal's right hand side is not unique
- **Common**
  - common subexpressions
- **Permuted**
  - permutations in commutative combinators
- **Lookalike**
  - near-clones
Spaghetti

- Uncluster
- Unsequence
- StartInTheMiddle

Mixture

- BadLayout
- Preprocessor

Shortage

- AlmostAlphabet
- ConfusingEntry
- Dead
- Bottom
- Debt
Structure / Proxy

- **Chain**
  - right hand side is a nonterminal

- **Throwaway**
  - a nonterminal used only once

- **Weak**
  - a nonterminal does not form a proper abstraction

- **Ghost**
  - a subexpression that should have been a nonterminal

- **Multitool**
  - a nonterminal that represents two or more concepts
/ Structure / Dependency

- **Diamond**
- **Rivalry**
  - overlapping alternatives
- **Ouroboros**
  - a circular dependency among modules
- **Soulmates**
  - nonterminals or modules are always changed together
- **Spillover**
  - a nonterminal that is too linked to the adjacent symbols
- **Mythic**
  - an alternative that is never exercised in the codebase
reference-type ::= class-type | interface-type
   | delegate-type | ... ;

class-type ::= type-name | "object" | "string";

interface-type ::= type-name ;

delegate-type ::= type-name ;
/ Structure / Complexity

- **TooWide**
  - right hand side length too big
- **TooRamose**
  - too much branching
- **TooRecursive**
  - too much recursion
- **TooNested**
  - groups are too deeply nested
- **TooTall**
  - distance from the root too big
- **Lonely**
  - large non-modular grammar

- **TooModular**
  - too many modules
- **Greedy**
  - one module that does too much
- **Lazy**
  - vice versa
- **TooCoupled**
  - low cohesion, high coupling
What do we know about smells?

- Smells are poor solutions to recurring problems [1]
- Smells are in between design and implementation [1]
- System experts detect smells worse than outsiders [2]
- Detection should be based on metrics [3]
- Systems with smells are worse [*]
- Effect of smells is negligible compared to size [4]
- Trial-and-error maintenance leads to smells [5]
- Devs rely on smell detectors in unfamiliar contexts [6]

[6] Sousa, Oliveira, Garcia, Lee, Conte, Oizumi, de Mello, Lopes, Valentim, Oliveira, Lucena, SBES'17
It is our responsibility to learn more
http://slebok.github.io/grass