Stefano Bellinelli

stefano.bellinelli@eoc.ch

Cosa è

Oggetto con troppe responsabilità

- «Quella classe mastodontica che poi fa tutto lui» (Collega anonimo)
- Troppi metodi
- Metodi non correlati
- Troppi attributi

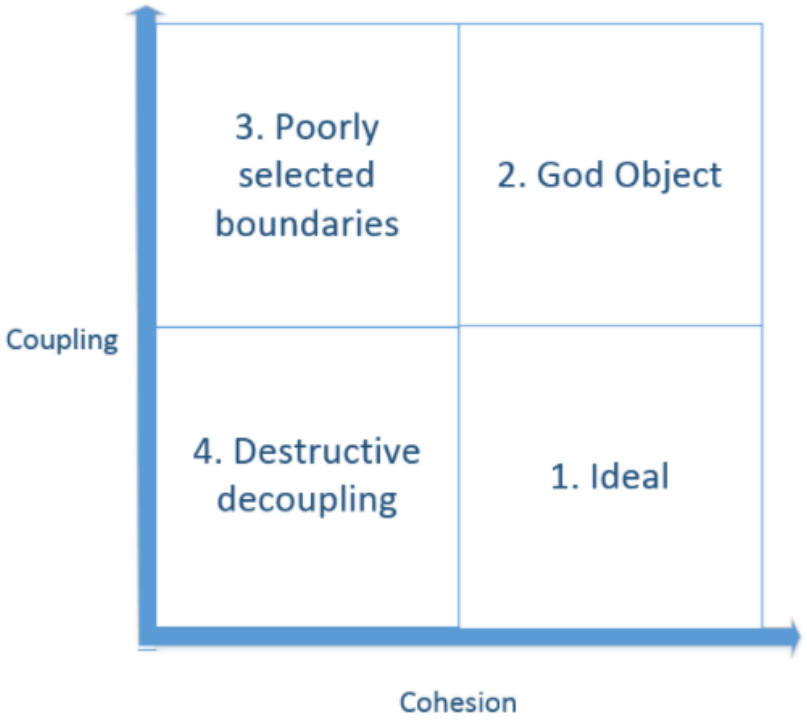
Coesione e accoppiamento

- **Coesione**: grado di unità all'interno di un modulo
- **Accoppiamento**: grado di dipendenza tra moduli

Coesione e accoppiamento (2)

Obiettivo:

- Molta coesione
- Poco accoppiamento



Coesione e accoppiamento (3)

1.

2.

3.

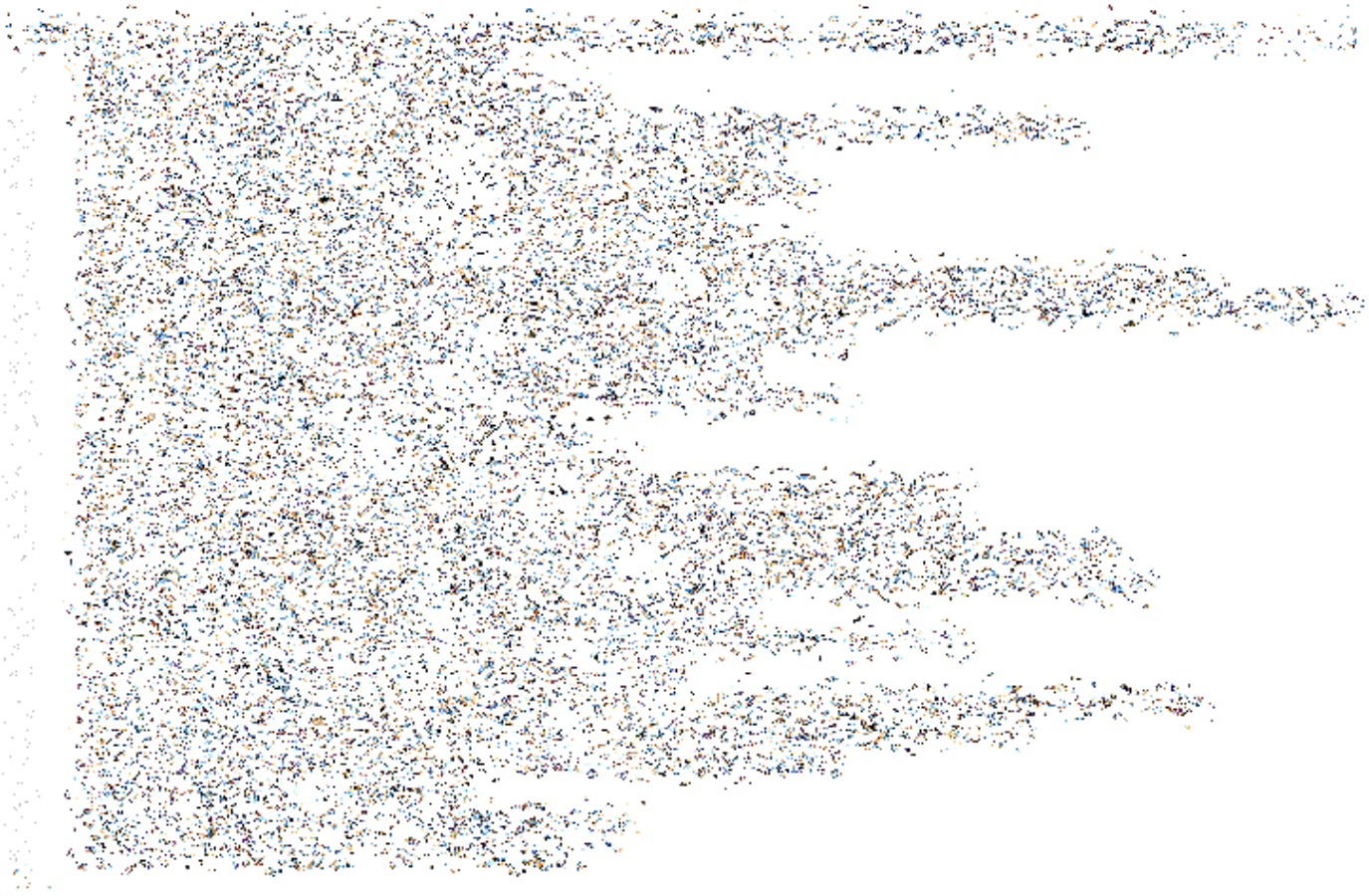
4.

Un piccolo esempio

ServiceGenerateService.java

- 650 righe
- 25 services @Autowired
- 5 metodi pubblici
- 3 metodi protected
- 15 metodi privati
- Metodi con 8 parametri, tra cui classi e primitive

Un costruttore...



Che problemi causa?

- Modifiche pericolose

- Difficile da testare

- Difficile passaggio di conoscenze

Come riconoscerlo

Può essere individuata da molti code smell:

- Large class
- Long method
- Long parameter list
- Data clumps
- Divergent change (modifica della classe per motivi diversi)

…

Calisthenics coinvolti

- Keep all entities small
- No classes with more than two instance variables

- One level of indentation per method (long method)

…

Come lo risolviamo/evitiamo?

Purtroppo pregare non basta

«Quando ero piccolo pregavo ogni notte per avere una bicicletta nuova.
Poi ho capito che il Signore non fa questo genere di cose, allora ne ho rubata una e gli ho chiesto di perdonarmi.»

(Emo Philips)

Come lo risolviamo?

- Creare una suite di test
- Identificare le chiamate ai metodi pubblici
- Dividere la classe in classi più piccole
- Identificare i metodi statici ed estrarli
- Raggruppare metodi e properties in classi
- Eliminare (o deprecare) la God Class

Come lo evitiamo?

- Single responsibility principle
- Regola del boy scout

Single responsibility principle

Single responsibility principle (2)

- Ogni elemento deve avere una sola responsabilità

«Gather together the things that change for the same reasons. Separate those things that change for different reasons.»

(Robert C. Martin)

Regola del boy scout

“Always check a module in cleaner than when you checked it out”
(Robert C. Martin)

Conclusioni

- Anti pattern
- Se non affrontato, peggiora con il tempo
- Rallenta enormemente lo sviluppo
- Scoraggia aggiornamenti e refactoring
- Va risolto un passo alla volta

Domande?

Thanks for watching!

Sources:

- Agile technical practices distilled (P.M. Santos, M. Consolaro, A. Di Gioia)
- Design Patterns Past and Future, Aleksandar Bulajic
- Programming Enlightenment, Robert C. Martin

- https://blog.cleancoder.com/
- https://it.wikipedia.org/wiki/Test_driven_development
- https://elliotchance.medium.com/the-transformation-priority-premise-tpp-3e5dc08d445e
- https://www.theserverside.com/tip/How-to-refactor-the-God-object-antipattern
- https://enterprisecraftsmanship.com/posts/cohesion-coupling-difference/

Stefano Bellinelli - stefano.bellinelli@eoc.ch