Connascence

“Two software components are connascent if a change in one would require the other to be modified in order to maintain the overall correctness of the system.”

[https://en.wikipedia.org/wiki/Connascence]

Is Connascence a concept that only exists in software development?

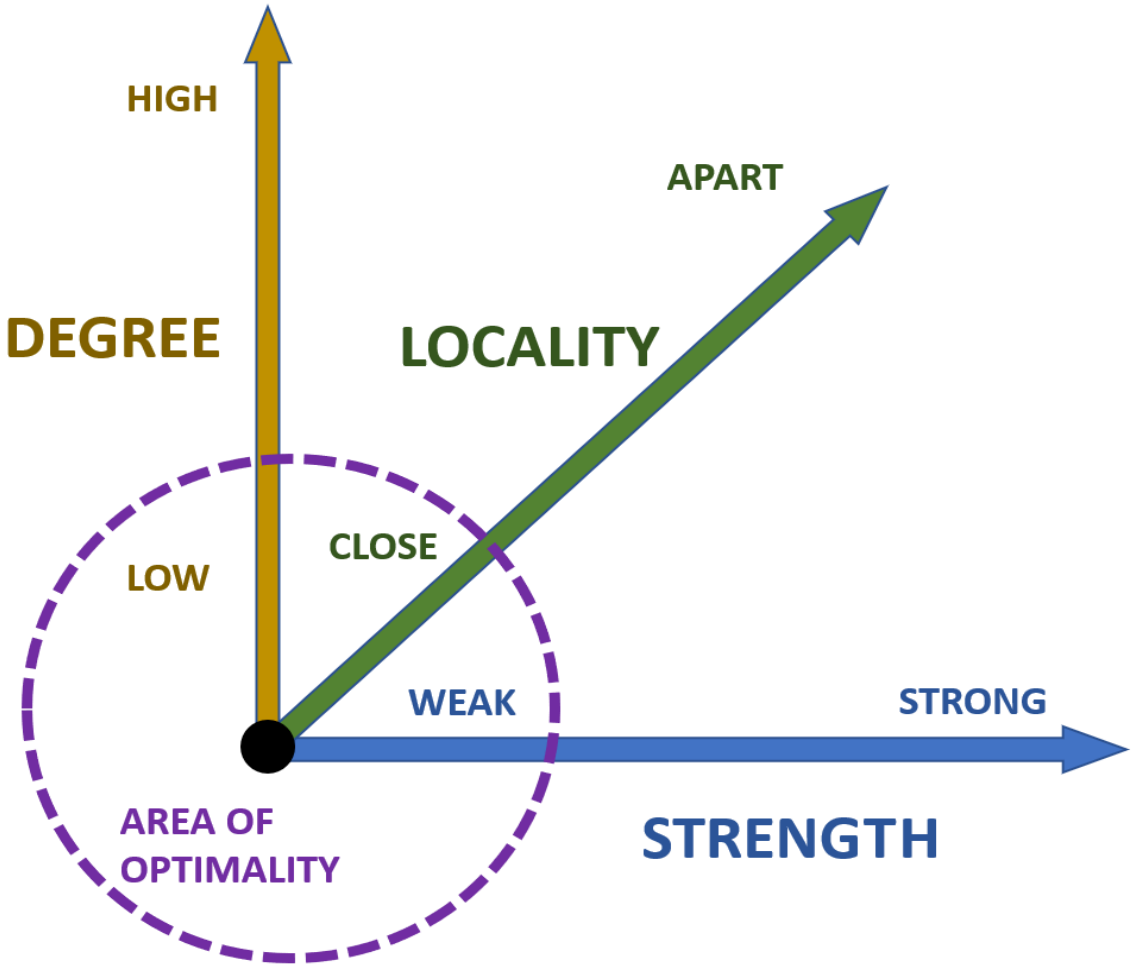
I don’t think so!

Connascence…

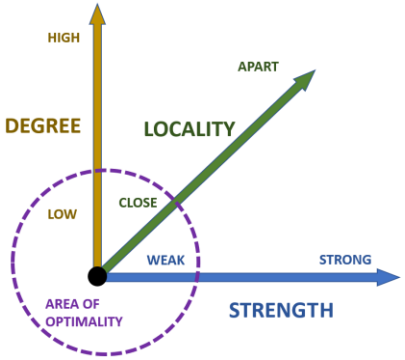
- was introduced by Meilir Page-Jones in 1992 (Comparing techniques by means of encapsulation and connascence, Communications of the ACM volume 35 issue 9, September 1992)
- is a software quality metric for measuring Coupling in OO-Systems
- is a metric you can consult during refactoring
- provides a classification using a taxonomy
- and hence allows reasoning about the complexity caused by Coupling

[Codemania 2015: Josh Robb - Connascence & Coupling]

The 3 Dimensions of Connascence



[Agile Technical Practices Distilled. Pedro M. Santos, Marco Consolaro, Alessandro Di Gioia]



Dimensions of Connascence: Degree

Degree is the size of the impact as estimated by the number of occurrences and the number of entities it affects

The acceptability of Connascence is strongly related to its degree: the higher the degree, the higher the pain when a modification is needed

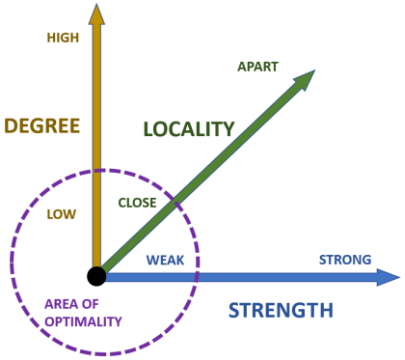
[Agile Technical Practices Distilled. Pedro M. Santos, Marco Consolaro, Alessandro Di Gioia]

The fewer entities involved, the better…

Ripple effect…

… shotgun surgery in politics…

Connascence – as any other metric – can‘t be seen in isolation!

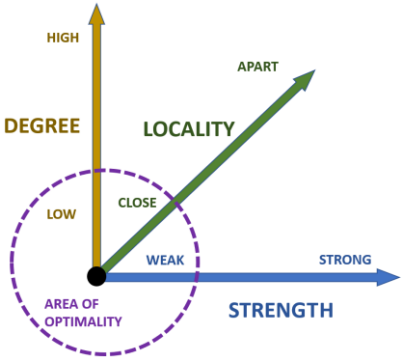


Dimensions of Connascence: Locality

The locality dimension deals with the closeness of the connascent elements in terms of abstraction (function, class, module)

For elements that are close together Connascence is more acceptable and to a certain degree desirable since this leads to Cohesion

On the other hand, the forms of Connascence between elements that are far apart should be weaker



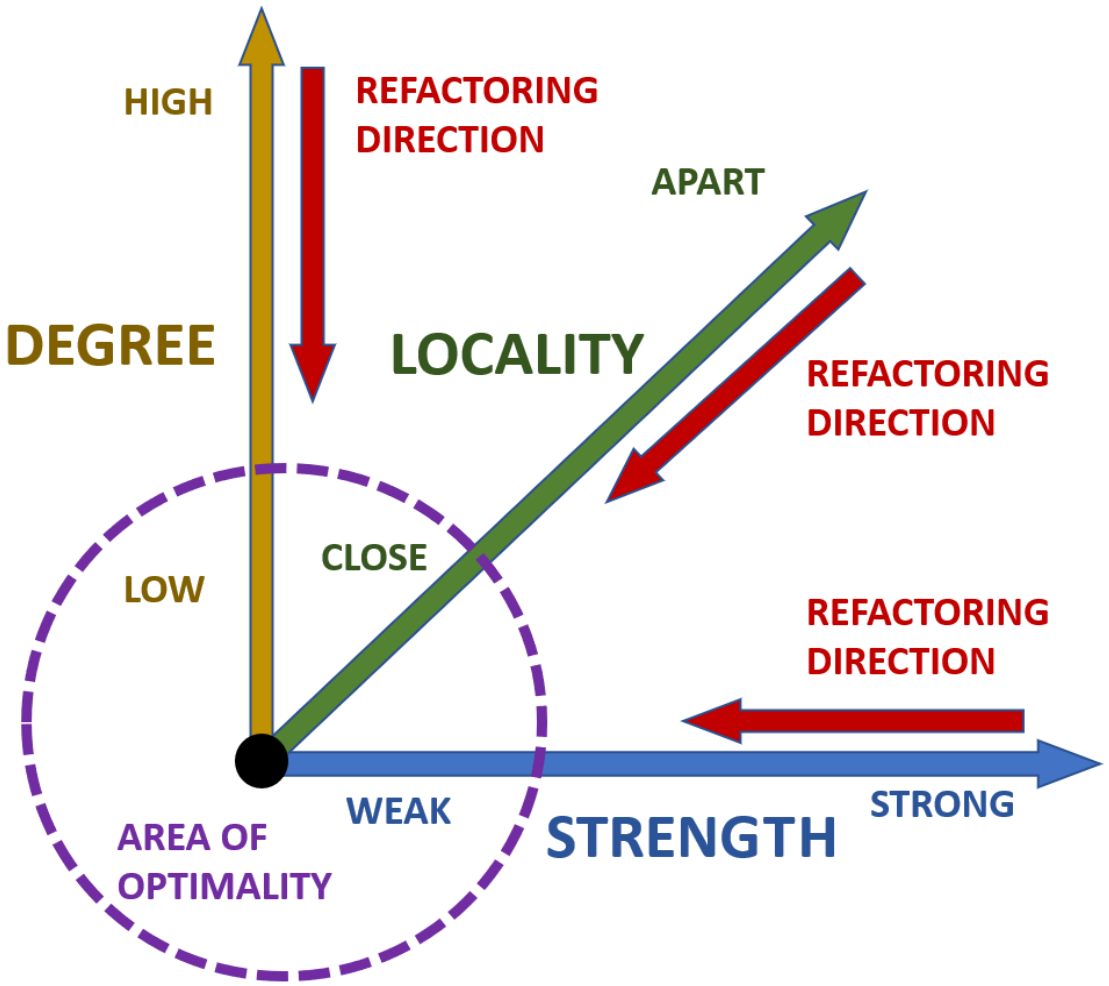
Dimensions of Connascence: Strength

A form of connascence is considered to be stronger if it is more likely to require compensating changes in connascent elements

The stronger the form of connascence, the more difficult and costly it is to change the elements in the relationship

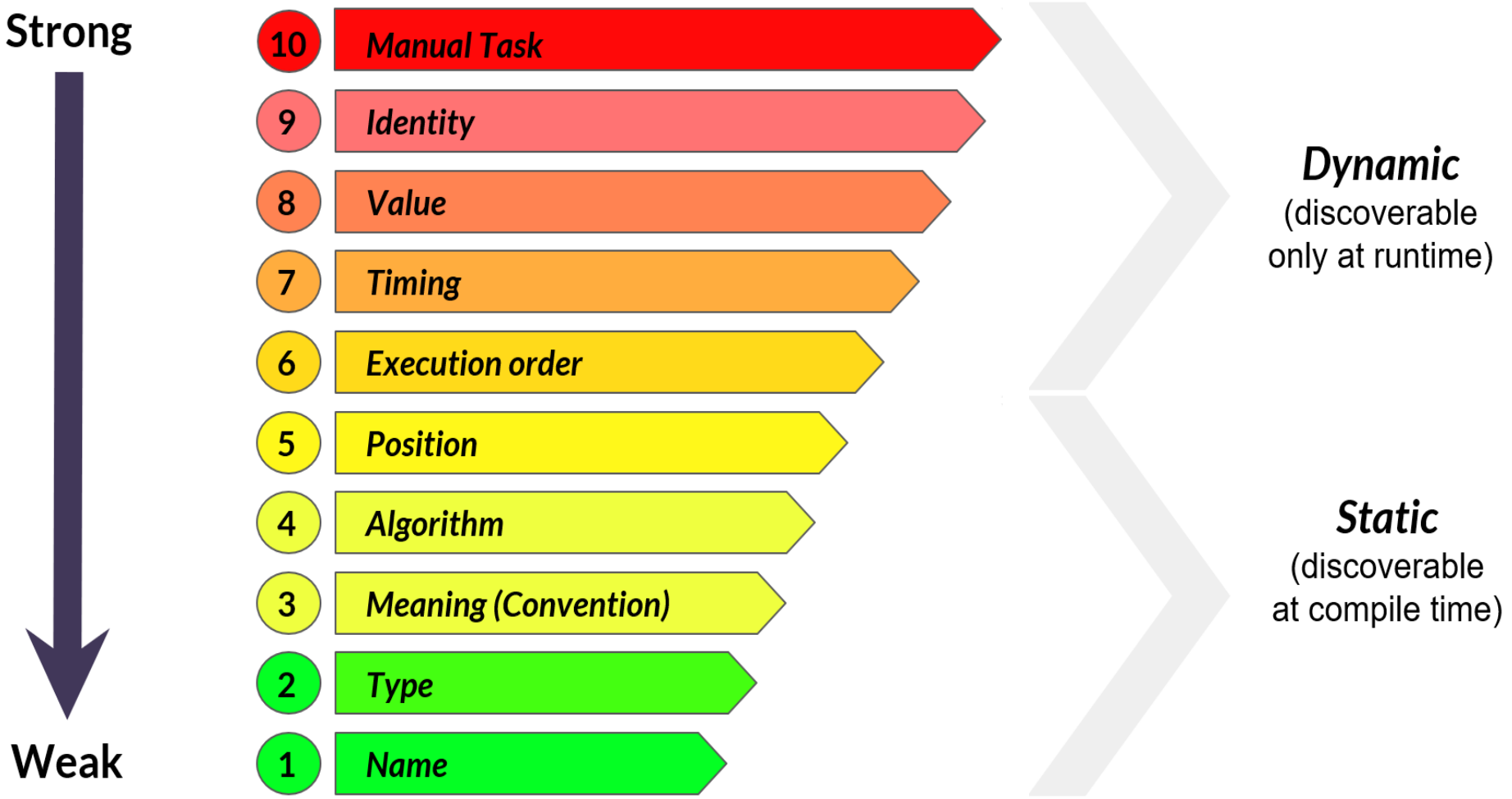
[https://en.wikipedia.org/wiki/Connascence]

The direction of refactoring



[Agile Technical Practices Distilled. Pedro M. Santos, Marco Consolaro, Alessandro Di Gioia]

Types of Connascence (Strength)



[Agile Technical Practices Distilled. Pedro M. Santos, Marco Consolaro, Alessandro Di Gioia]

A real world examle: calcuating the BMI

```java
class ClassWithBmi {
    String bmi;
    // other fields
    public void setBmi(String bmi) {
        this.bmi = bmi;
    }
    public String getBmi() {
        return bmi;
    }
    // other methods
}
```

```java
class ClassUsingTheBmi {
    ClassWithBmi bmiClass;
    // ..someFields
    void someMethod(){
        bmiClass = new ClassWithBmi();
        bmiClass.setBmi(SomeHelperClass.calculateBmi_NameUnrelatedToParameters("c1", "c2"));
    }

    void someOtherMethod(){
        final String bmi = bmiClass.getBmi();
        // do something with the bmi
    }
}
```

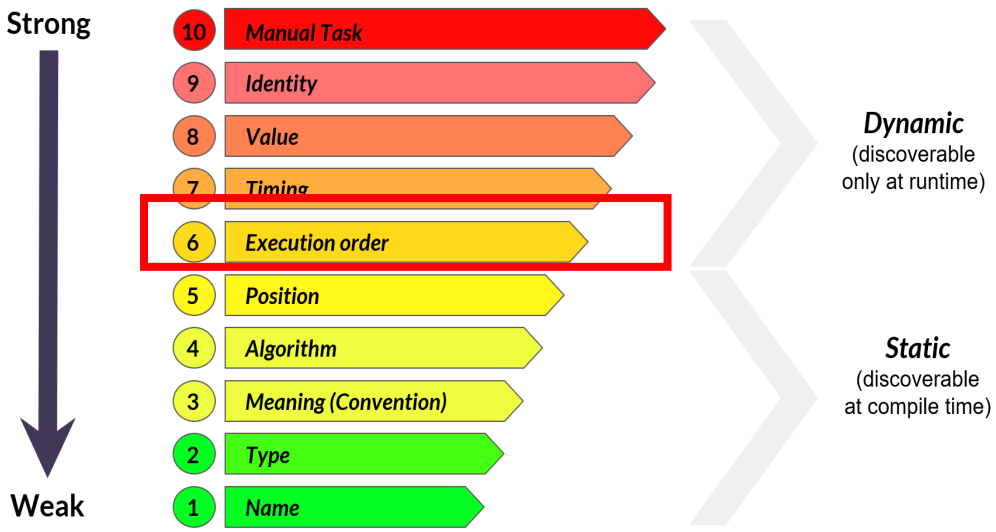
```java
class SomeHelperClass {
    static SomeService someService;
    public static String calculateBmi_NameUnrelatedToParameters(String cryptic1, String cryptic2) {
        SomeDataClass data = someService.getPersonRelatedData(cryptic1, cryptic2);
        final String calculatedBmi = "--> do some calculation with data";
        return calculatedBmi;
    }
}
```

Phase 1: Stating the problem(s)

```java
class ClassWithBmi {
    String bmi;
    // other fields
    public void setBmi(String bmi) {
        this.bmi = bmi;
    }
    public String getBmi() {
        return bmi;
    }
    // other methods
}
```

- Connascence of Execution Order
- Low Cohesion
- Data/Lazy class
- Primitive obsession
- …

Objects of this class are invalid after being created!

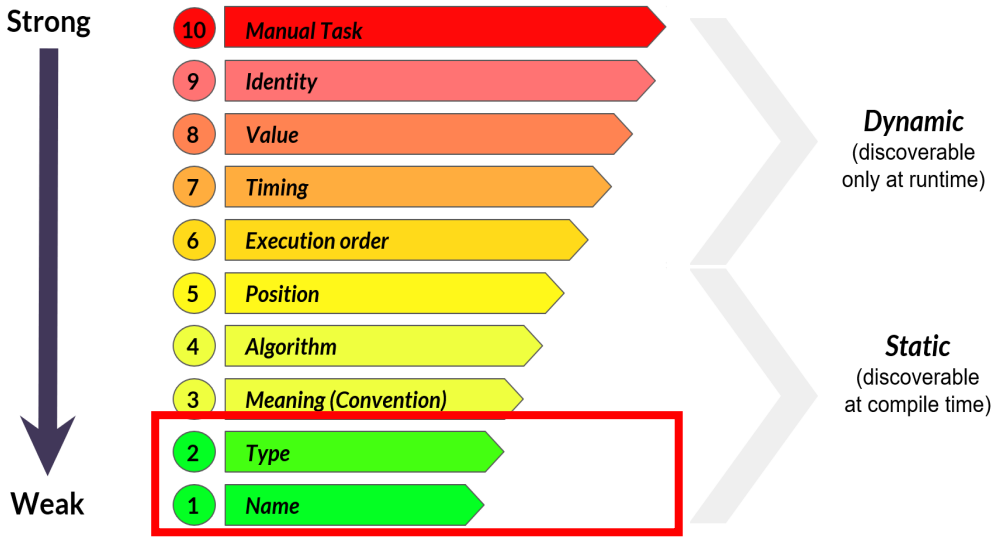


Phase 2: Make the world a better place

```java
class Person {
    Bmi bmi;
    // other fields
    public Person(double size, double weight/* maybe other*/){
        this.bmi = new Bmi(size, weight);
        //...
    }
    void doSomethingWithTheBmi() {
        // do Something with the bmi
    }
    // other methods
}
```

```java
class Bmi {
    double bmi;
    public Bmi(final double size, final double weight) {
        bmi = calculateBmi(size, weight);
    }
    private double calculateBmi(final double size, final double weight) {
        return // formula here
    }
    public boolean isTooLow(){
        return // evaluate bmi
    }
    ...
}
```

- Connascence of Name and Type
- High Cohesion
- Code smells are gone



Phase 2: Make the world a better place

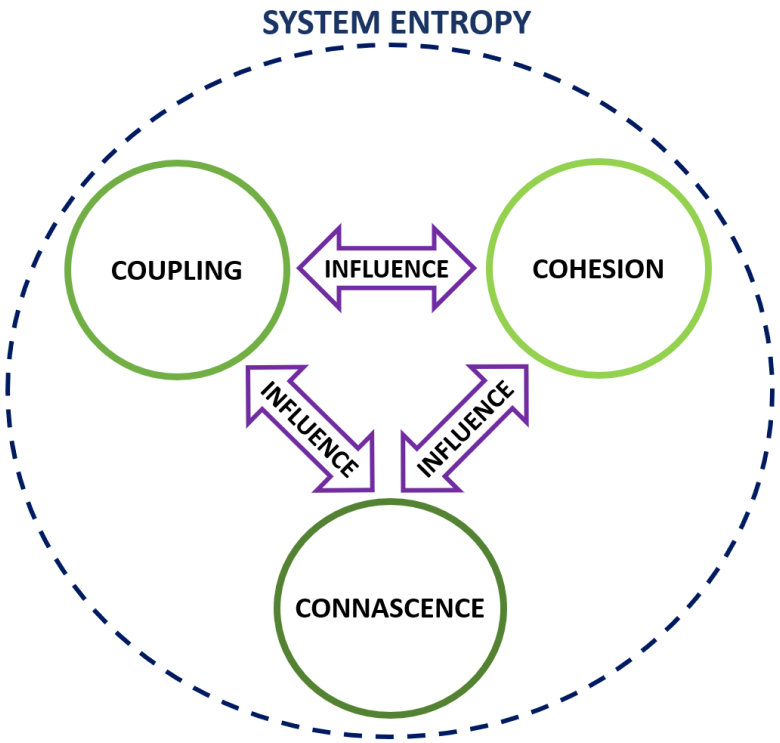
```
class SomeHelperClass {
    static SomeService someService;
    public static String calculateBmi_NameUn[illegible]edToParameters(String cryptic1, String cryptic2) {
        SomeDataClass data = someService.c[illegible]ers[illegible]elatedData(cryptic1, cryptic2);
        final String calculatedBmi = "[illegible] do some c[illegible]lation with data";
        return calculatedBmi;
    }
}
```

Calcuating the BMI: the whole truth!

- The Algorithm is implemented in 2 independently deployed software modules!
- Therefore we have Connascence of Algorithm (4) with a high locality factor
- One module is very feature envious, but we won't get money nor time to combine them in the near future

So, what would you do? Are there any suggestions for what would be an intermediate step to partially fix this issue?

So where does Connascence fit in?



Entropy can be seen as…

- The energy dispersal rate of the system
- The degree of disorder in a system
- An increasing function of the number of possible states of a system

Our goal must be to keep it as low as possible!

Cohesion, Coupling and Connascence are gauges for entropy

[Agile Technical Practices Distilled. Pedro M. Santos, Marco Consolaro, Alessandro Di Gioia]

?

Thank you for your attention!

Contact: philipp.eichenberger@css.ch