On the Evolution of Inheritance and Delegation Mechanisms and Their impact on Code Quality

Giammaria Giordano, Antonio Fasulo, Gemma Catolino, Fabio Palomba, Filomena Ferrucci, Carmine Gravino

giagiordano@unisa.it, a.fasulo@student.unisa.it gcatolinotilburguniversity.edu, fpalomba@unisa.it, fferrucci@unisa.it, cgravino@unisa.it

Introduction

Software reusability refers to the development practice through which developers make use of existing code when implementing new functionalities.
Object-Oriented (OO) programming languages ,e.g., JAVA, provide developers with various mechanisms supporting code reusability: examples are design patterns, the use of third-party libraries, and programming abstractions.
Focusing on JAVA, there are two well-known abstraction mechanisms such as **inheritance** and **delegation**.
Inheritance is the process by which one class takes the property of another class: the new classes, known as derived or children's classes, inherit the attributes and/or the behavior of the pre-existing classes, which are referred to as base, super, or parent classes.
Delegation is, instead, the mechanism through which a class uses an object instance of another class by forwarding it messages and letting it performing actions.

In addition, the sub-optimal adoption of inheritance and delegation mechanisms had led to the definition and investigation of reusability-specific code smells.

This paper builds on this line of research by proposing an empirical analysis of how inheritance and delegation mechanisms evolve over time as well as their effects of software quality evolution. More particularly, we first mine evolutionary data pertaining to 15 releases of three open-source projects. Then, we statistically compare the number of inheritance and delegation mechanisms implemented over subsequent releases in order to assess the trend followed by the adoption of those metrics. Finally, we build a statistical model relating inheritance and delegation metrics, as well as other confounding factors, to the variation of code smell severity, in an effort of understanding the impact of reusability metrics on the likelihood of code smells to become more/less severe over time.

Methodology

The goal of the empirical study was to assess how inheritance and delegation mechanisms evolve over time and how they impact the severity of code smells during software evolution, with the purpose of understanding the extent to which reusability mechanisms applied by developers may provide indications on the future quality of source code.
RQ1.How do developers adopt source code reusability mechanisms during software evolution?
RQ2.How do source code reusability mechanisms impact the severity of code smells over time?
The context of the empirical study consisted of 15 releases of three JAVA systems such as JHOTDRAW, APACHE ANT, and JEDIT
The first step of our empirical study was concerned with the computation of reusability metrics and, specifically, the adoption of specification inheritance, implementation inheritance, and delegation.
RQ1: we analysed the distributions of the three metrics denoting the reuse—Inhimpl, Inhspec, and Del— to understand how these evolve over time.
For each subsequent release, Ri and Rj, we applied non-parametric statistical tests to verify whether the distribution of each reusability metric differed between Ri and Rj. First, we applied the Mann-Whitney test: the choice was due to the sample size and the non-normality of the distributions considered. Second, we complemented the analysis with the application of the Cliff's Delta (δ), and finally we normalized the values of the reusability metrics by LOC. The results were intended to be statistically significant α= 0.05.
RQ2. we defined a statistical model relating reusability metrics and other control factors to the increase/decrease of code smell severity. we first selected the actual code smell types investigated. These were: Good Class, Spaghetti Code, Complex Class, and Class Data Should be Private. We computed the differences between the actual metric values and the corresponding thresholds used by DECOR;(2) We normalized the obtained differences in the range [0;1] using the min-max strategy; and (3) We computed the final severity score as the mean of the normalized values. Given the nature of our categorical response variable, i.e., the categories "decrease", "stable", and "increase", we used a Multinomial Log Linear model to study the severity of the four code smells considered.

Research Questions

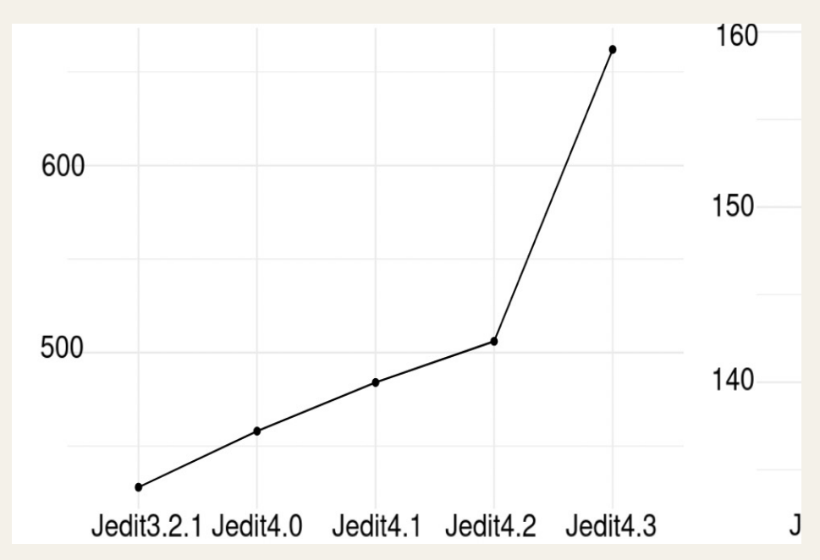
RQ1.1. How does source code reuse in terms of implementation inheritance vary in software evolution?

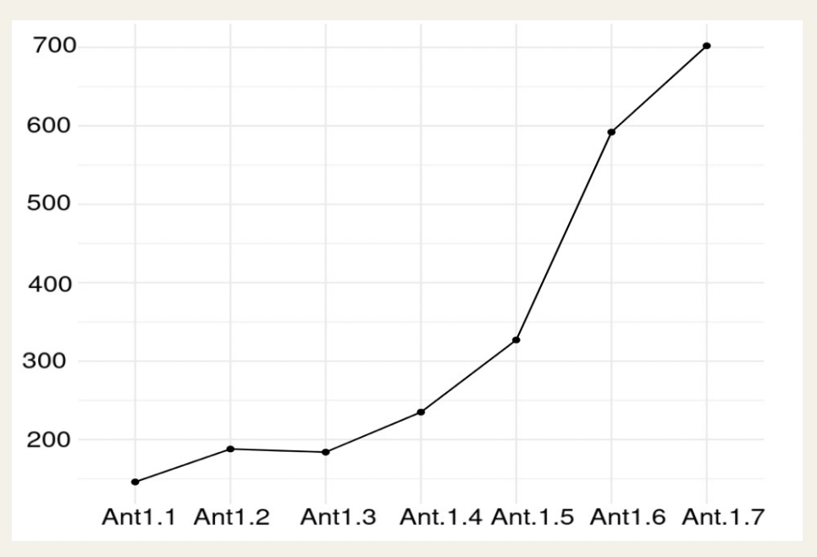
RQ1.2. How does reuse in terms of specification inheritance vary in software evolution?

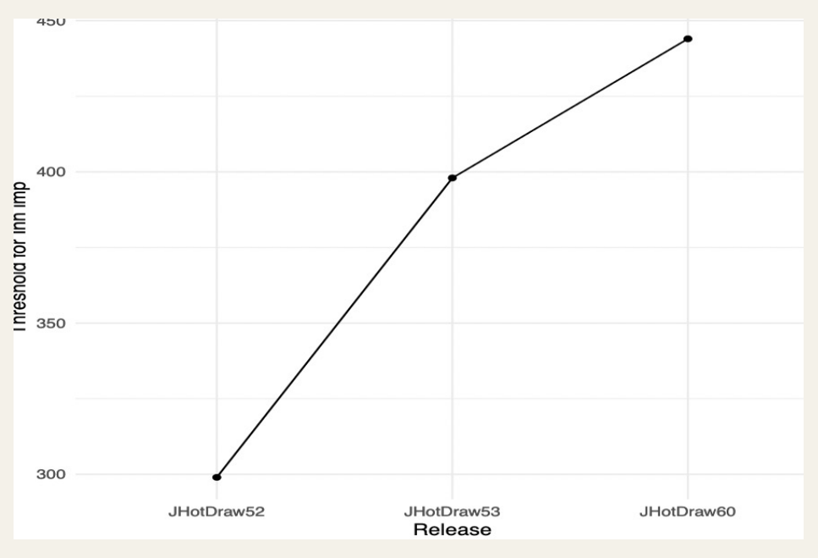
RQ1.3. How does reuse in terms of delegation vary in software evolution?

RQ2. How do source code reusability mechanisms impact the severity of code smells over time?

Results RQ1.1



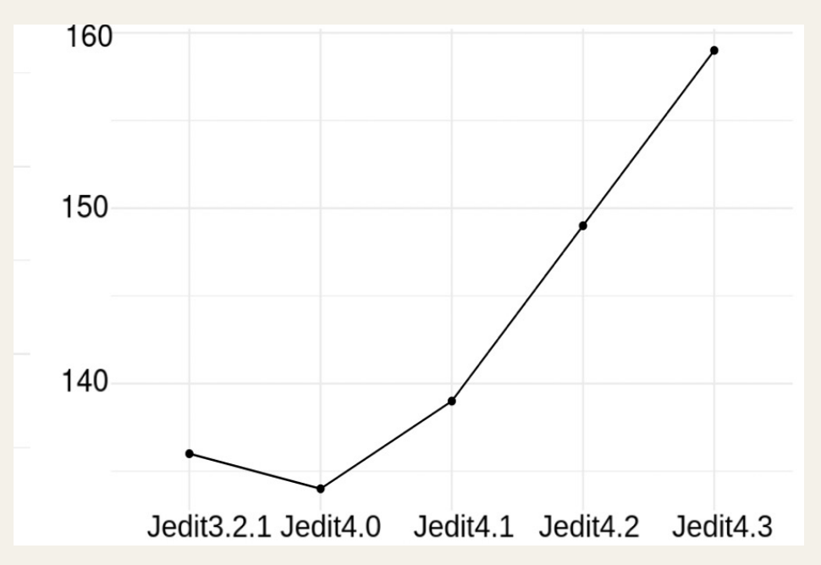


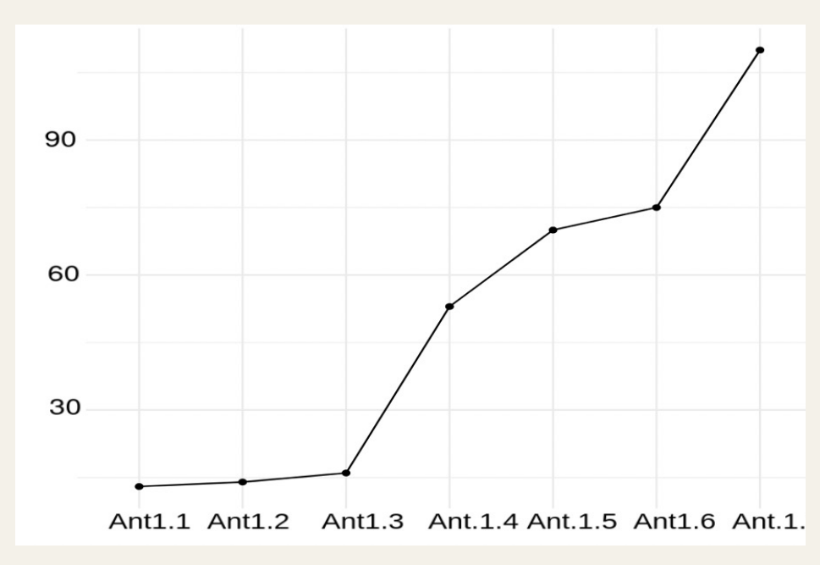


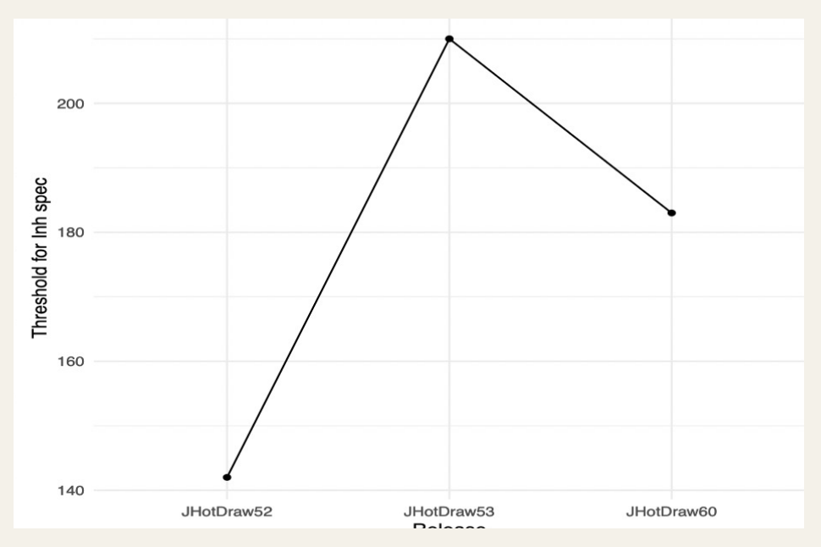
Mann-Whitney test and Cliff's Delta (δ) was applied

RQ11.The use of implementation inheritance increases over time, even if not in a statistically significant manner

Results RQ.1.2



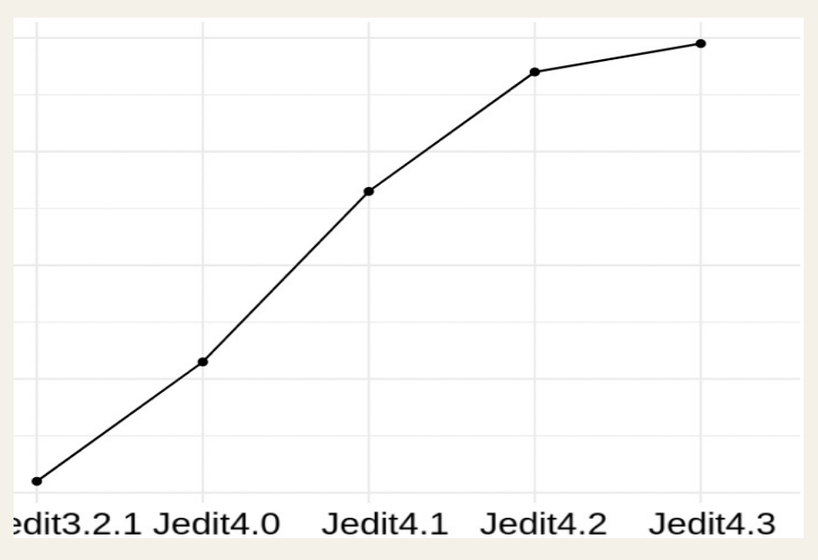


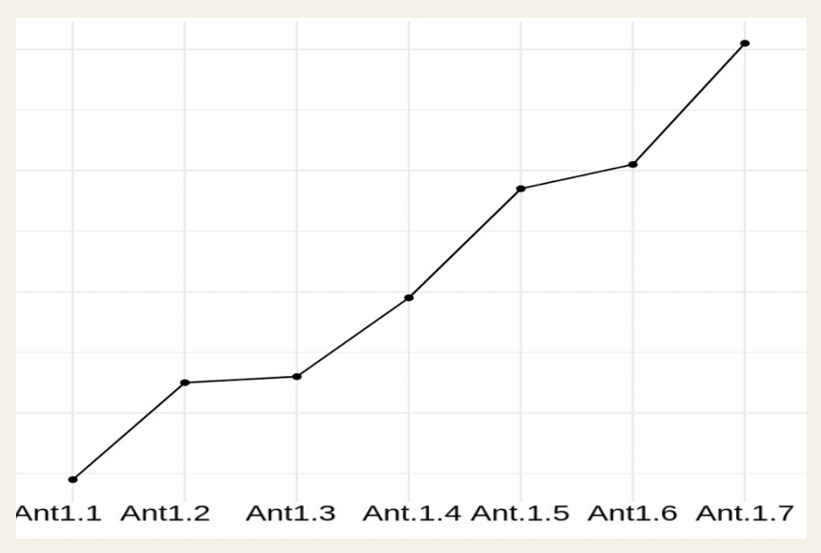


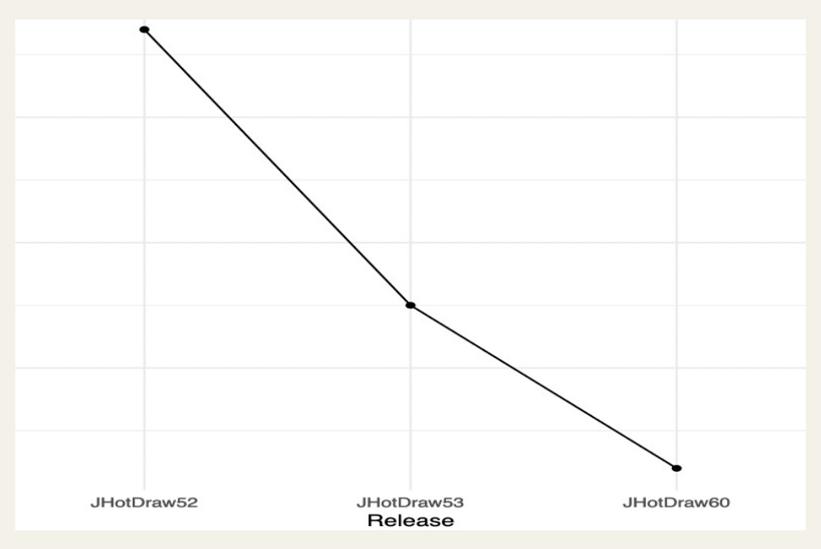
Mann-Whitney test and Cliff's Delta (δ) was applied

RQ.1.2 The adoption of specification inheritance is stable over time. The only exception was JHOTDRAW, that preferred to use different reusability mechanisms while defining a new milestone

Results RQ. 1.3



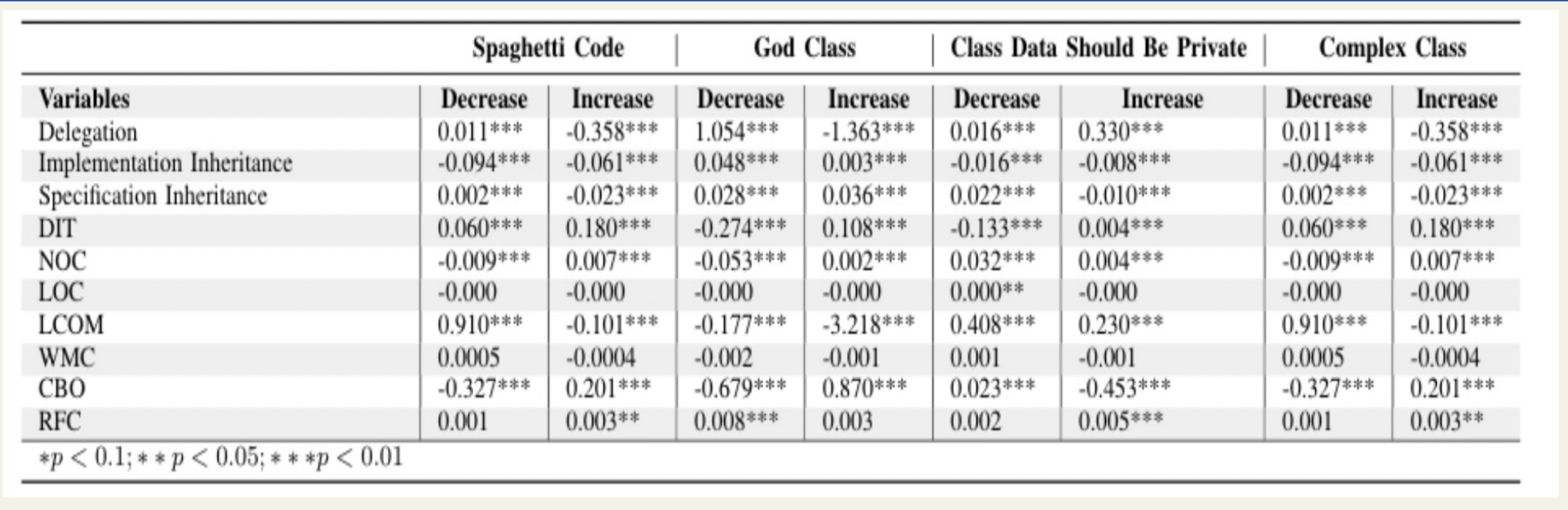




Mann-Whitney test and Cliff's Delta (δ) was applied

RQ. 1.3. The adoption of delegations increases over time, but not in a statistically significant way. The exception is the one of JHOTDRAW, where the trend observed suggests a progressive reluctance to this mechanism which might be worth of studying in the future.

Results RQ. 2



	Spaghetti Code		God Class		Class Data Should Be Private		Complex Class	
Variables	Decrease	Increase	Decrease	Increase	Decrease	Increase	Decrease	Increase
Delegation	0.011***	-0.358***	1.054***	-1.363***	0.016***	0.330***	0.011***	-0.358***
Implementation Inheritance	-0.094***	-0.061***	0.048***	0.003***	-0.016***	-0.008***	-0.094***	-0.061***
Specification Inheritance	0.002***	-0.023***	0.028***	0.036***	0.022***	-0.010***	0.002***	-0.023***
DIT	0.060***	0.180***	-0.274***	0.108***	-0.133***	0.004***	0.060***	0.180***
NOC	-0.009***	0.007***	-0.053***	0.002***	0.032***	0.004***	-0.009***	0.007***
LOC	-0.000	-0.000	-0.000	-0.000	0.000**	-0.000	-0.000	-0.000
LCOM	0.910***	-0.101***	-0.177***	-3.218***	0.408***	0.230***	0.910***	-0.101***
WMC	0.0005	-0.0004	-0.002	-0.001	0.001	-0.001	0.0005	-0.0004
CBO	-0.327***	0.201***	-0.679***	0.870***	0.023***	-0.453***	-0.327***	0.201***
RFC	0.001	0.003**	0.008***	0.003	0.002	0.005***	0.001	0.003**

$*p < 0.1; **p < 0.05; ***p < 0.01$

It is important to recognize that the decrease of the CK metric values— considered as control variables in our models — correlate well with the decreasing of code smell intensity.

RQ. 2. In most cases, delegation and inheritance metrics positively correlate to the decrease of the code smell severity. Nonetheless, in some cases, their presence causes instability

Discussion

Developers tend to increase the use of inheritance and delegation over time: this may suggest that code reuse is indeed are levant matter for developers, which may be worth of further investigations. Nonetheless, such an increase does not turn to be statistically significant.

⇒ The research community should invest some effort in understanding the developer's perspective on the use of abstraction mechanisms.

In most cases, the correct adoption of inheritance and delegation positively related to the decrease of code smell severity.

⇒ Inheritance and delegation mechanisms can be used by practitioners as instruments to decrease the severity of code smells. Researchers might be interested in devising novel semi-automated tools that might support practitioners in employing these abstraction mechanisms.

In some cases, we observed that inheritance and delegation may lead to instability ,i.e., they simultaneously increase and decrease the code smell severity.

⇒ Researchers should consider the definition of qualitative or mixed-method studies through which understand what are the boundaries between a correct and incorrect use of inheritance and delegation mechanisms.

Conclusion

In this paper, we investigated the evolution of reusability mechanisms and their impact on the variation of code smell severity. Our results revealed that the adoption of inheritance and delegation increases over time, even though not in a statistically significant manner. At the same time, such an evolution does have an impact on code smells: our statistical approach found inheritance and delegation metrics to significantly impact the likelihood of the severity of the code smells considered to vary. Often, such a variation can be considered positive, meaning that a proper adoption of inheritance and delegation reduces the severity of code smells.

Acknowledgements

Gemma is partially supported by the European Commission grant no. 825040 (RADON). Fabio is supported by the Swiss National Science Foundation through the SNF Project No.PZ00P2 186090 (TED)

Contact Information

Giammaria Giordano

Email: giagiordano@unisa.it
broke31.github.io/giammaria-giordano/
https://www.instagram.com/broke31sf/
https://it.linkedin.com/in/giammaria-giordano