Stefano Lambiase

slambiase@unisa.it

Software Project Management

Recent Research on Anti-Patterns and Their Impact

Today's Lecture

- Introduction to measurement and metrics in software project management
- Focus on social debt
- Impact of social debt on technical debt
- Predicting socio-technical debt
- Tools to support the measurement of social debt

Software Project Management, why does it matter?

70%

of software projects

FAIL

Jim Johnson, *“CHAOS 2020: Beyond Infinity.”*, Standish Group (2020).

Factors that influence project outcome

- Project under-estimated 81%
- Bad risk management 75%
- Bad development team management 67%
- **Human and social aspects 75%**
- …

Cerpa Narciso and June M. Verner, *“Why Did Your Project Fail?”*, Communications of the ACM, 2009

Factors that influence project outcome

- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]

Most of the factors that influence project outcomes are related to collaboration and communication between stakeholders.

In other words, they concern the spectrum of software project management responsibilities.

Cerpa Narciso and June M. Verner, *“Why Did Your Project Fail?”*, Communications of the ACM, 2009

Software development has never been so distributed!

Stray and Moe

“Understanding coordination in global software engineering: A mixed-methods study on the use of meetings and Slack”

Journal of Systems and Software, 2020

Software development has never been so distributed!

Software Project Management

Distributed Software Project Management

Stray and Moe

"Understanding coordination in global software engineering: A mixed-methods study on the use of meetings and Slack"

Journal of Systems and Software, 2020

Software development has never been so distributed!

Stray and Moe

"Understanding coordination in global software engineering: A mixed-methods study on the use of meetings and Slack"

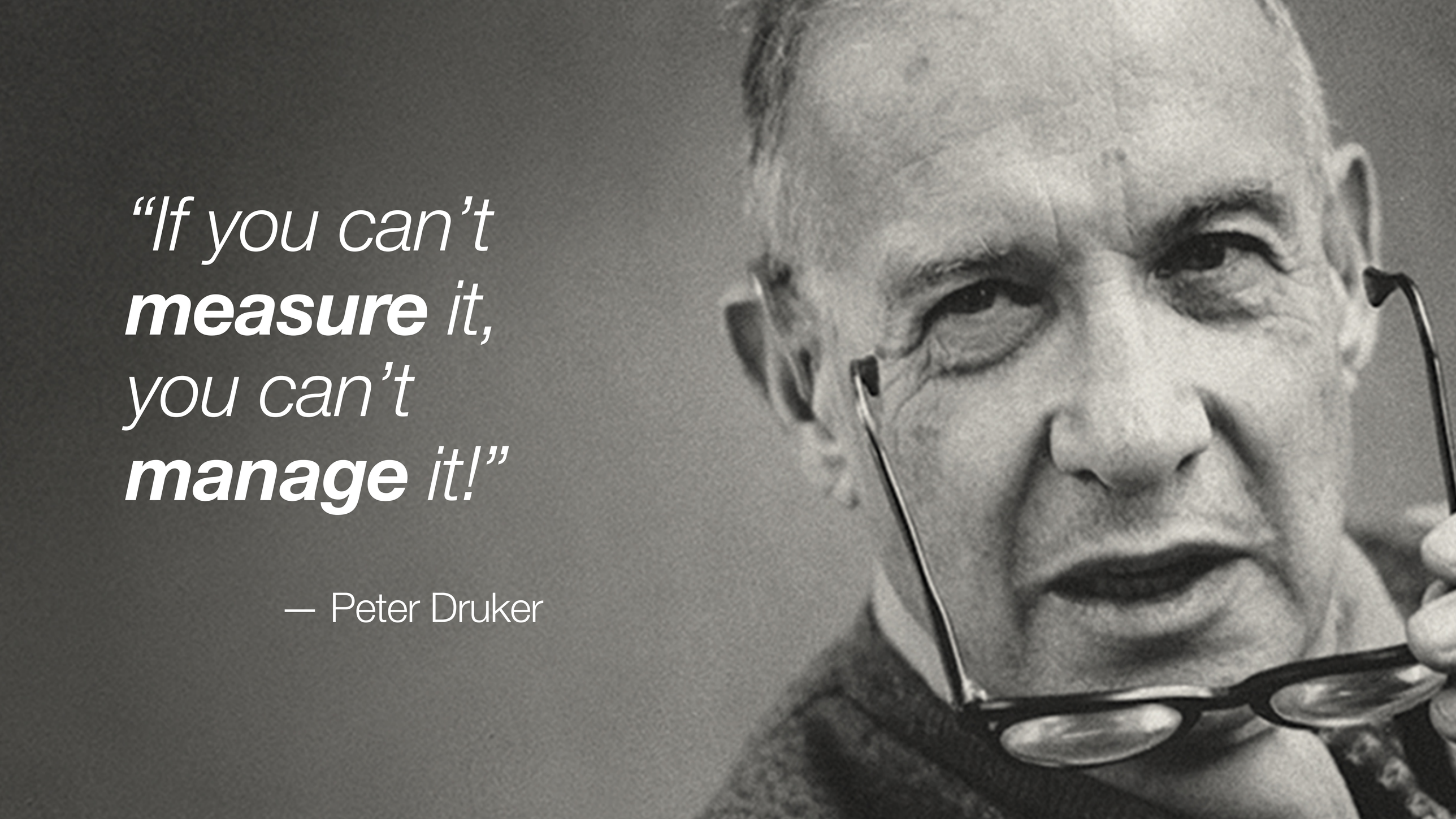
Journal of Systems and Software, 2020

Limitation!

Researchers tend to treat social aspects in a **qualitative** and **abstract** way

“If you can’t ***measure*** *it, you can’t* ***manage*** *it!”*

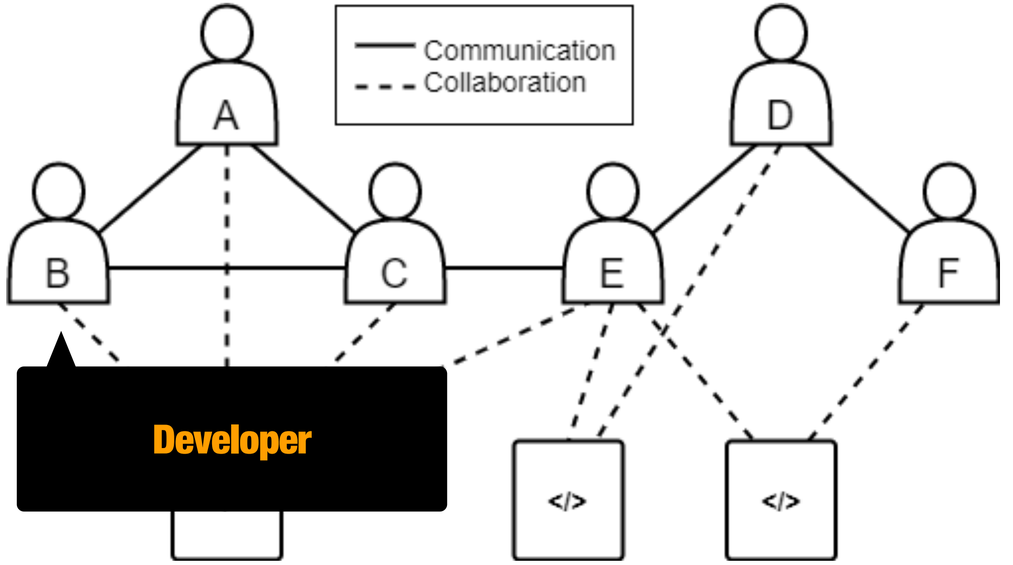
— Peter Druker



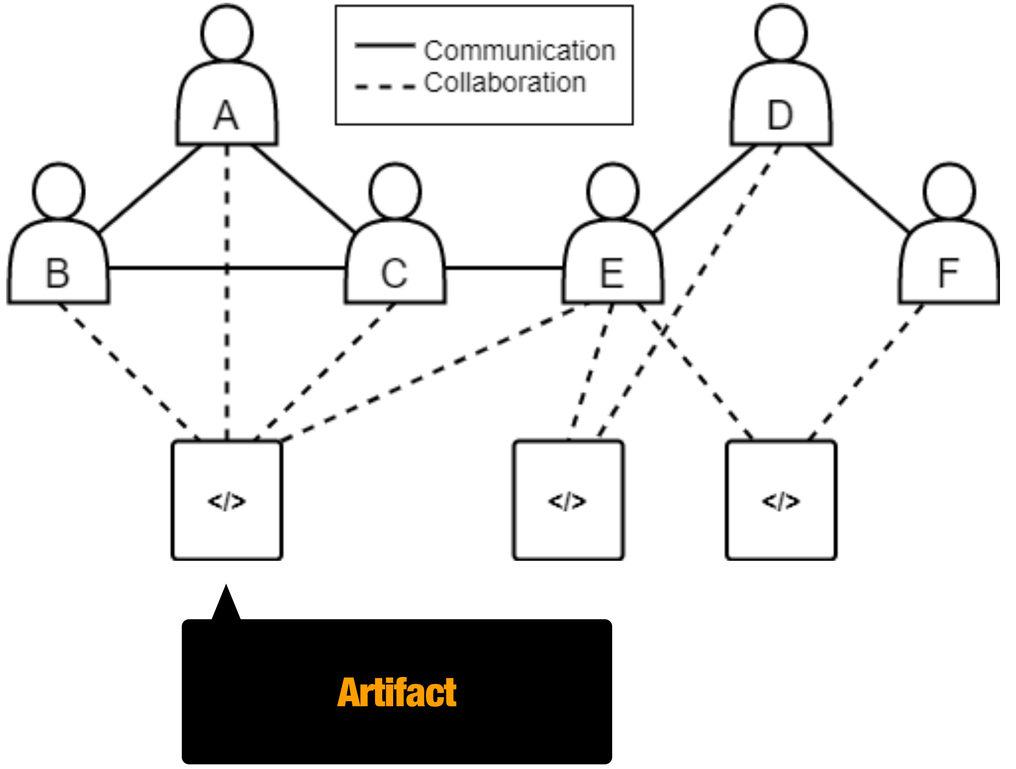
Software Project Management, How to measure?

Category	Metric	Description
Developer Social Network metrics	devs	Number of developers present in the global Developers Social Network
	ml.only.devs	Number of developers present only in the communication Developers Social Network
	code.only.devs	Number of developers present only in the collaboration Developers Social Network
	ml.code.devs	Number of developers present both in the collaboration and in the communication DSNs
	perc.ml.only.devs	Percentage of developers present only in the communication Developers Social Network
	perc.code.only.devs	Percentage of developers present only in the collaboration Developers Social Network
	perc.ml.code.devs	Percentage of developers present both in the collaboration and in the communication DSNs
	sponsored.devs	Number of sponsored developers (95% of their commits are done in working hours)
	ratio.sponsored	Ratio of sponsored developers with respect to developers present in the collaboration DSN
Socio-Technical Metrics	st.congruence	Estimation of socio-technical congruence
	communicability	Estimation of information communicability (decisions diffusion)
	num.tz	Number of timezones involved in the software development
	ratio.smelly.devs	Ratio of developers involved in at least one Community Smell
Core community members metrics	core.global.devs	Number of core developers of the global Developers Social Network
	core.mail.devs	Number of core developers of the communication Developers Social Network
	core.code.devs	Number of core developers of the collaboration Developers Social Network
	sponsored.core.devs	Number of core sponsored developers
	ratio.sponsored.core	Ratio of core sponsored developers with respect to core developers of the collaboration DSN
	global.truck	Ratio of non-core developers of the global Developers Social Network
	mail.truck	Ratio of non-core developers of the communication Developers Social Network
	code.truck	Ratio of non-core developers of the collaboration Developers Social Network
	mail.only.core.devs	Number of core developers present only in the communication DSN
	code.only.core.devs	Number of core developers present only in the collaboration DSN
	ml.code.core.devs	Number of core developers present both in the communication and in the collaboration DSNs
	ratio.mail.only.core	Ratio of core developers present only in the communication DSN
	ratio.code.only.core	Ratio of core developers present only in the collaboration DSN
	ratio.ml.code.core	Ratio of core developers present both in the communication and in the collaboration DSNs
Turnover	global.turnover	Global developers turnover with respect to the previous temporal window
	code.turnover	Collaboration developers turnover with respect to the previous temporal window
	core.global.turnover	Core global developers turnover with respect to the previous temporal window
	core.mail.turnover	Core communication developers turnover with respect to the previous temporal window
	core.code.turnover	Core collaboration developers turnover with respect to the previous temporal window
	ratio.smelly.quitters	Ratio of developers previously involved in any Community Smell that left the community
Social Network Analysis metrics	closeness.centr	SNA degree metric of the global DSN computed using closeness
	betweenness.centr	SNA degree metric of the global DSN computed using betweenness
	degree.centr	SNA degree metric of the global DSN computed using degree
	global.mod	SNA modularity metric of the global DSN
	mail.mod	SNA modularity metric of the communication Developers Social Network
	code.mod	SNA modularity metric of the collaboration Developers Social Network
	density	SNA density metric of the global Developers Social Network

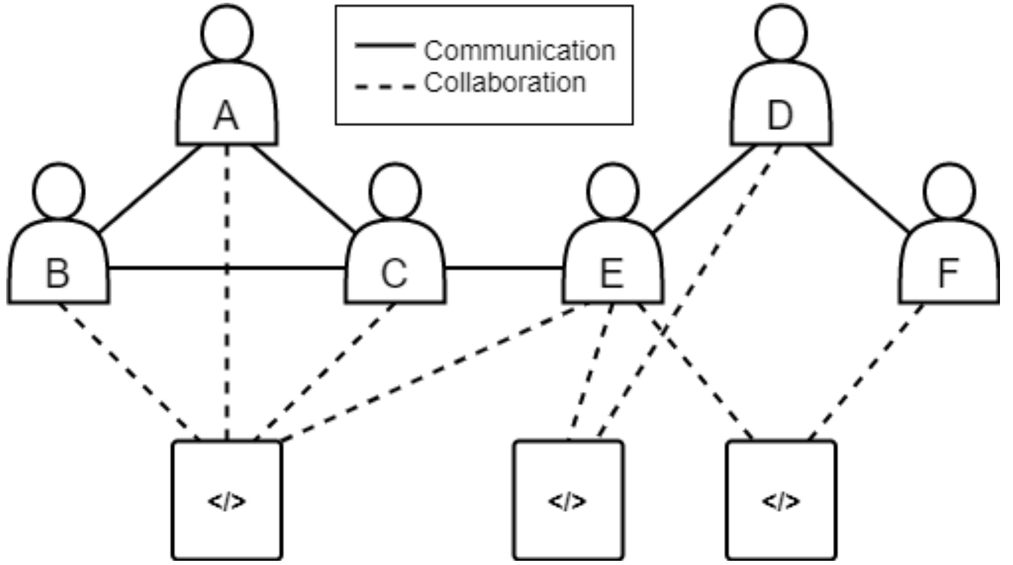
elopers present in the global Developers Social Network
elopers present only in the communication Developers Social Network
elopers present only in the collaboration Developers Social Network
elopers present both in the collaboration and in the communication DSNs
levelopers present only in the communication Developers Social Network
levelopers present only in the collaboration Developers Social Network
levelopers present both in the collaboration and in the communication DSNs
nsored developers (95% of their commits are done in working hours)
ored developers with respect to developers present in the collaboration DSN
ocio-technical congruence
nformation communicability (decisions diffusion)
ezones involved in the software development
pers involved in at least one Community Smell
e developers of the global Developers Social Network
e developers of the communication Developers Social Network
e developers of the collaboration Developers Social Network
e sponsored developers
ponsored developers with respect to core developers of the collaboration DSN
re developers of the global Developers Social Network
re developers of the communication Developers Social Network
re developers of the collaboration Developers Social Network
e developers present only in the communication DSN
e developers present only in the collaboration DSN
e developers present both in the communication and in the collaboration DSNs
evelopers present only in the communication DSN
evelopers present only in the collaboration DSN
evelopers present both in the communication and in the collaboration DSNs
ers turnover with respect to the previous temporal window
levelopers turnover with respect to the previous temporal window
velopers turnover with respect to the previous temporal window
cation developers turnover with respect to the previous temporal window
ion developers turnover with respect to the previous temporal window
pers previously involved in any Community Smell that left the community
etric of the global DSN computed using closeness
etric of the global DSN computed using betweenness
etric of the global DSN computed using degree
y metric of the global DSN
y metric of the communication Developers Social Network
y metric of the collaboration Developers Social Network
etric of the global Developers Social Network



elopers present in the global Developers Social Network
elopers present only in the communication Developers Social Network
elopers present only in the collaboration Developers Social Network
elopers present both in the collaboration and in the communication DSNs
levelopers present only in the communication Developers Social Network
levelopers present only in the collaboration Developers Social Network
levelopers present both in the collaboration and in the communication DSNs
nsored developers (95% of their commits are done in working hours)
ored developers with respect to developers present in the collaboration DSN
ocio-technical congruence
nformation communicability (decisions diffusion)
ezones involved in the software development
pers involved in at least one Community Smell
e developers of the global Developers Social Network
e developers of the communication Developers Social Network
e developers of the collaboration Developers Social Network
e sponsored developers
ponsored developers with respect to core developers of the collaboration DSN
re developers of the global Developers Social Network
re developers of the communication Developers Social Network
re developers of the collaboration Developers Social Network
e developers present only in the communication DSN
e developers present only in the collaboration DSN
e developers present both in the communication and in the collaboration DSNs
evelopers present only in the communication DSN
evelopers present only in the collaboration DSN
evelopers present both in the communication and in the collaboration DSNs
ers turnover with respect to the previous temporal window
levelopers turnover with respect to the previous temporal window
velopers turnover with respect to the previous temporal window
cation developers turnover with respect to the previous temporal window
ion developers turnover with respect to the previous temporal window
pers previously involved in any Community Smell that left the community
etric of the global DSN computed using closeness
etric of the global DSN computed using betweenness
etric of the global DSN computed using degree
y metric of the global DSN
y metric of the communication Developers Social Network
y metric of the collaboration Developers Social Network
etric of the global Developers Social Network

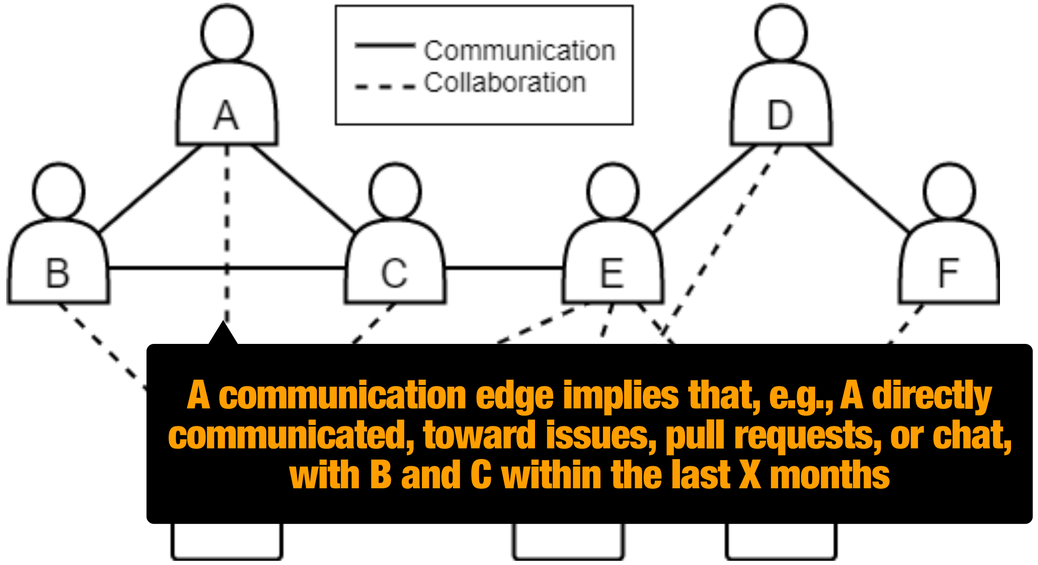


elopers present in the global Developers Social Network
elopers present only in the communication Developers Social Network
elopers present only in the collaboration Developers Social Network
elopers present both in the collaboration and in the communication DSNs
levelopers present only in the communication Developers Social Network
levelopers present only in the collaboration Developers Social Network
levelopers present both in the collaboration and in the communication DSNs
nsored developers (95% of their commits are done in working hours)
ored developers with respect to developers present in the collaboration DSN
ocio-technical congruence
nformation communicability (decisions diffusion)
ezones involved in the software development
pers involved in at least one Community Smell
e developers of the global Developers Social Network
e developers of the communication Developers Social Network
e developers of the collaboration Developers Social Network
e sponsored developers
ponsored developers with respect to core developers of the collaboration DSN
re developers of the global Developers Social Network
re developers of the communication Developers Social Network
re developers of the collaboration Developers Social Network
e developers present only in the communication DSN
e developers present only in the collaboration DSN
e developers present both in the communication and in the collaboration DSNs
evelopers present only in the communication DSN
evelopers present only in the collaboration DSN
evelopers present both in the communication and in the collaboration DSNs
ers turnover with respect to the previous temporal window
levelopers turnover with respect to the previous temporal window
velopers turnover with respect to the previous temporal window
cation developers turnover with respect to the previous temporal window
ion developers turnover with respect to the previous temporal window
pers previously involved in any Community Smell that left the community
etric of the global DSN computed using closeness
etric of the global DSN computed using betweenness
etric of the global DSN computed using degree
y metric of the global DSN
y metric of the communication Developers Social Network
y metric of the collaboration Developers Social Network
letric of the global Developers Social Network

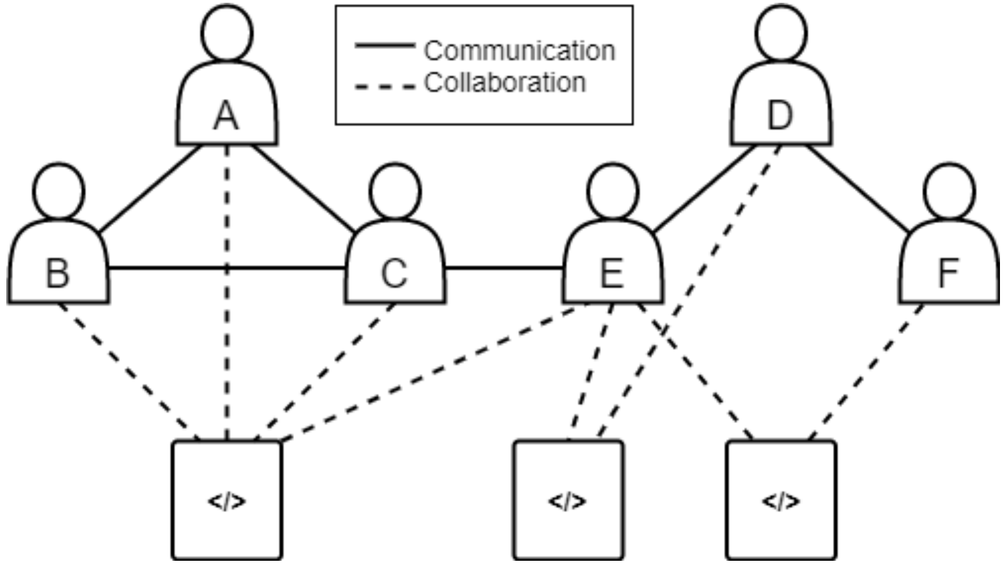


A collaboration edge implies that, e.g., A, B, C, and E co-committed changes on the same artifact within the last X months

elopers present in the global Developers Social Network
elopers present only in the communication Developers Social Network
elopers present only in the collaboration Developers Social Network
elopers present both in the collaboration and in the communication DSNs
levelopers present only in the communication Developers Social Network
levelopers present only in the collaboration Developers Social Network
levelopers present both in the collaboration and in the communication DSNs
nsored developers (95% of their commits are done in working hours)
ored developers with respect to developers present in the collaboration DSN
ocio-technical congruence
nformation communicability (decisions diffusion)
ezones involved in the software development
pers involved in at least one Community Smell
e developers of the global Developers Social Network
e developers of the communication Developers Social Network
e developers of the collaboration Developers Social Network
e sponsored developers
ponsored developers with respect to core developers of the collaboration DSN
re developers of the global Developers Social Network
re developers of the communication Developers Social Network
re developers of the collaboration Developers Social Network
e developers present only in the communication DSN
e developers present only in the collaboration DSN
e developers present both in the communication and in the collaboration DSNs
evelopers present only in the communication DSN
evelopers present only in the collaboration DSN
evelopers present both in the communication and in the collaboration DSNs
ers turnover with respect to the previous temporal window
levelopers turnover with respect to the previous temporal window
velopers turnover with respect to the previous temporal window
cation developers turnover with respect to the previous temporal window
ion developers turnover with respect to the previous temporal window
pers previously involved in any Community Smell that left the community
etric of the global DSN computed using closeness
etric of the global DSN computed using betweenness
etric of the global DSN computed using degree
y metric of the global DSN
y metric of the communication Developers Social Network
y metric of the collaboration Developers Social Network
letric of the global Developers Social Network

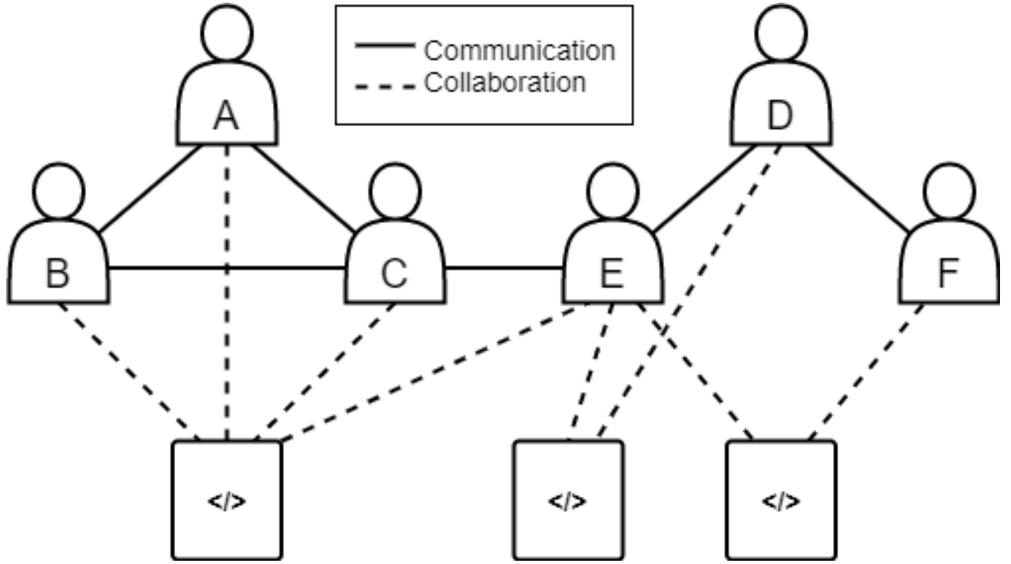


elopers present in the global Developers Social Network
elopers present only in the communication Developers Social Network
elopers present only in the collaboration Developers Social Network
elopers present both in the collaboration and in the communication DSNs
levelopers present only in the communication Developers Social Network
levelopers present only in the collaboration Developers Social Network
levelopers present both in the collaboration and in the communication DSNs
nsored developers (95% of their commits are done in working hours)
ored developers with respect to developers present in the collaboration DSN
ocio-technical congruence
nformation communicability (decisions diffusion)
ezones involved in the software development
pers involved in at least one Community Smell
e developers of the global Developers Social Network
e developers of the communication Developers Social Network
e developers of the collaboration Developers Social Network
e sponsored developers
ponsored developers with respect to core developers of the collaboration DSN
re developers of the global Developers Social Network
re developers of the communication Developers Social Network
re developers of the collaboration Developers Social Network
e developers present only in the communication DSN
e developers present only in the collaboration DSN
e developers present both in the communication and in the collaboration DSNs
evelopers present only in the communication DSN
evelopers present only in the collaboration DSN
evelopers present both in the communication and in the collaboration DSNs
ers turnover with respect to the previous temporal window
levelopers turnover with respect to the previous temporal window
velopers turnover with respect to the previous temporal window
cation developers turnover with respect to the previous temporal window
ion developers turnover with respect to the previous temporal window
pers previously involved in any Community Smell that left the community
etric of the global DSN computed using closeness
etric of the global DSN computed using betweenness
etric of the global DSN computed using degree
y metric of the global DSN
y metric of the communication Developers Social Network
y metric of the collaboration Developers Social Network
etric of the global Developers Social Network



Turnover: In developers' social graphs, turnover refers to the rate at which developers join and leave a particular community, team, or network over a given period. This concept is crucial in understanding the dynamics within a developer community, as **high turnover can disrupt social bonds, collaborative workflows, and knowledge sharing.**

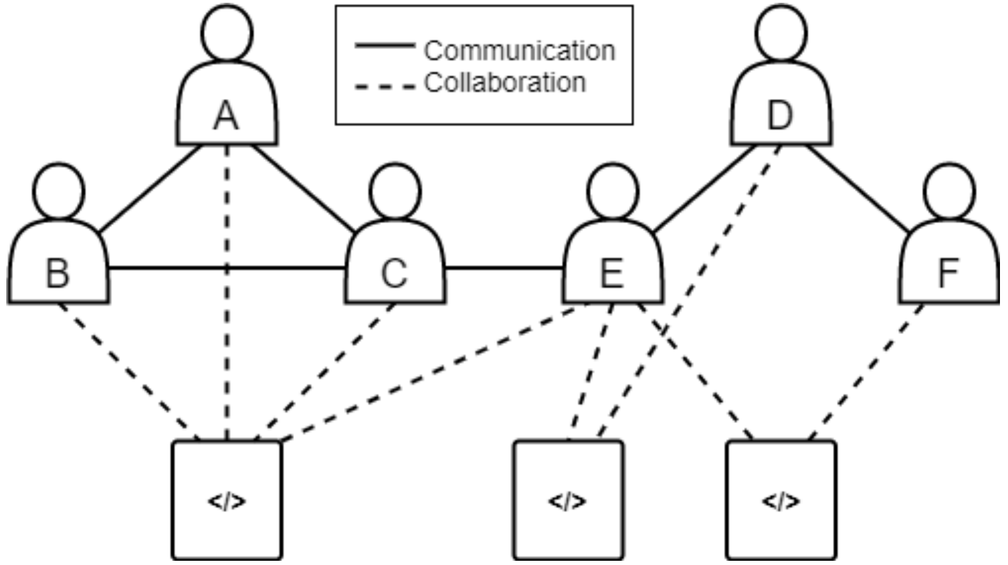
elopers present in the global Developers Social Network
elopers present only in the communication Developers Social Network
elopers present only in the collaboration Developers Social Network
elopers present both in the collaboration and in the communication DSNs
levelopers present only in the communication Developers Social Network
levelopers present only in the collaboration Developers Social Network
levelopers present both in the collaboration and in the communication DSNs
nsored developers (95% of their commits are done in working hours)
ored developers with respect to developers present in the collaboration DSN
ocio-technical congruence
nformation communicability (decisions diffusion)
ezones involved in the software development
pers involved in at least one Community Smell
e developers of the global Developers Social Network
e developers of the communication Developers Social Network
e developers of the collaboration Developers Social Network
e sponsored developers
ponsored developers with respect to core developers of the collaboration DSN
re developers of the global Developers Social Network
re developers of the communication Developers Social Network
re developers of the collaboration Developers Social Network
e developers present only in the communication DSN
e developers present only in the collaboration DSN
e developers present both in the communication and in the collaboration DSNs
evelopers present only in the communication DSN
evelopers present only in the collaboration DSN
evelopers present both in the communication and in the collaboration DSNs
ers turnover with respect to the previous temporal window
levelopers turnover with respect to the previous temporal window
velopers turnover with respect to the previous temporal window
cation developers turnover with respect to the previous temporal window
ion developers turnover with respect to the previous temporal window
pers previously involved in any Community Smell that left the community
etric of the global DSN computed using closeness
etric of the global DSN computed using betweenness
etric of the global DSN computed using degree
y metric of the global DSN
y metric of the communication Developers Social Network
y metric of the collaboration Developers Social Network
etric of the global Developers Social Network



Turnover: In developers' social graphs, turnover refers to the rate at which developers join and leave a particular community, team, or network over a given period. This concept is crucial in understanding the dynamics within a developer community, as **high turnover can disrupt social bonds, collaborative workflows, and knowledge sharing.**

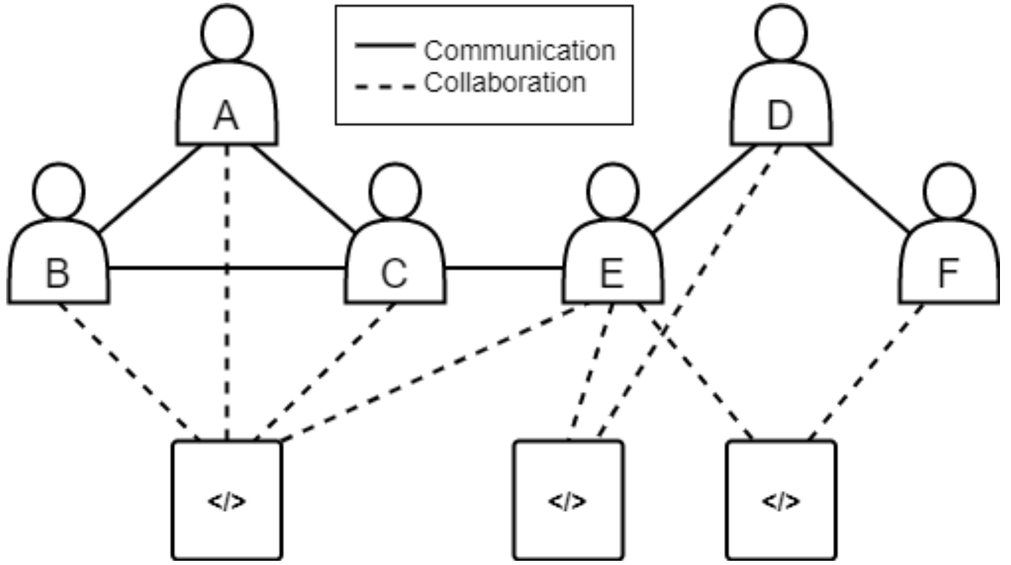
1. Node turnover: The number of nodes (i.e., developers) entering or exiting the social graph within a specific time frame.
2. Edge turnover: Changes in connections (edges) between developers, reflecting shifts in collaborative or communicative relationships.
3. Turnover rate: A quantifiable measure that reflects the ratio of developers leaving versus those joining.

elopers present in the global Developers Social Network
elopers present only in the communication Developers Social Network
elopers present only in the collaboration Developers Social Network
elopers present both in the collaboration and in the communication DSNs
levelopers present only in the communication Developers Social Network
levelopers present only in the collaboration Developers Social Network
levelopers present both in the collaboration and in the communication DSNs
nsored developers (95% of their commits are done in working hours)
ored developers with respect to developers present in the collaboration DSN
ocio-technical congruence
nformation communicability (decisions diffusion)
ezones involved in the software development
pers involved in at least one Community Smell
e developers of the global Developers Social Network
e developers of the communication Developers Social Network
e developers of the collaboration Developers Social Network
e sponsored developers
ponsored developers with respect to core developers of the collaboration DSN
re developers of the global Developers Social Network
re developers of the communication Developers Social Network
re developers of the collaboration Developers Social Network
e developers present only in the communication DSN
e developers present only in the collaboration DSN
e developers present both in the communication and in the collaboration DSNs
evelopers present only in the communication DSN
evelopers present only in the collaboration DSN
evelopers present both in the communication and in the collaboration DSNs
ers turnover with respect to the previous temporal window
levelopers turnover with respect to the previous temporal window
velopers turnover with respect to the previous temporal window
cation developers turnover with respect to the previous temporal window
ion developers turnover with respect to the previous temporal window
pers previously involved in any Community Smell that left the community
etric of the global DSN computed using closeness
etric of the global DSN computed using betweenness
etric of the global DSN computed using degree
y metric of the global DSN
y metric of the communication Developers Social Network
y metric of the collaboration Developers Social Network
letric of the global Developers Social Network



Truck Factor: Also known as the *bus factor*, is a measure of the risk within a software development project that arises from knowledge concentration. It represents the **minimum number of developers who**, **if incapacitated** (e.g., "hit by a truck"), **would cause the project to face severe difficulties in being completed or maintained.**

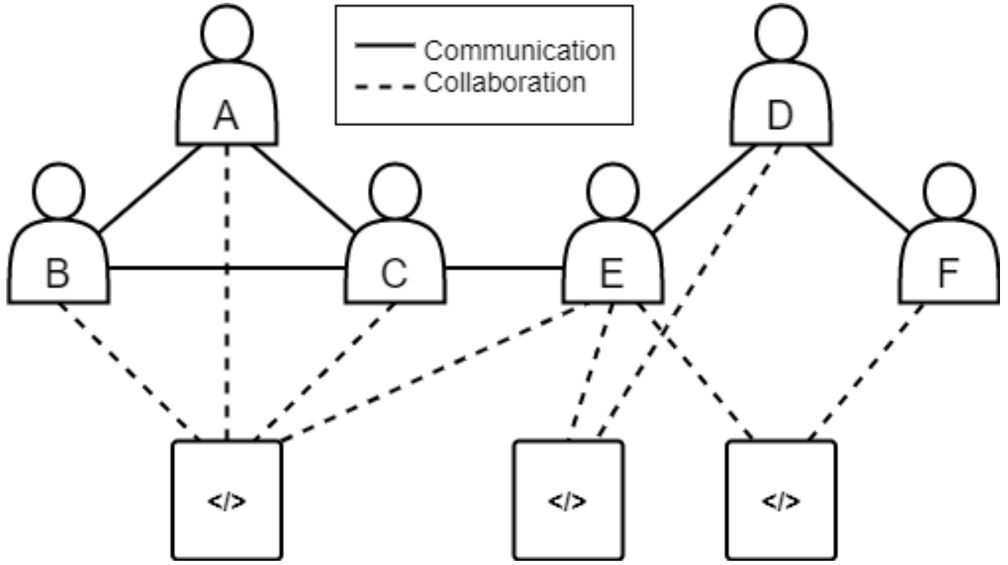
elopers present in the global Developers Social Network
elopers present only in the communication Developers Social Network
elopers present only in the collaboration Developers Social Network
elopers present both in the collaboration and in the communication DSNs
levelopers present only in the communication Developers Social Network
levelopers present only in the collaboration Developers Social Network
levelopers present both in the collaboration and in the communication DSNs
nsored developers (95% of their commits are done in working hours)
ored developers with respect to developers present in the collaboration DSN
ocio-technical congruence
nformation communicability (decisions diffusion)
ezones involved in the software development
pers involved in at least one Community Smell
e developers of the global Developers Social Network
e developers of the communication Developers Social Network
e developers of the collaboration Developers Social Network
e sponsored developers
ponsored developers with respect to core developers of the collaboration DSN
re developers of the global Developers Social Network
re developers of the communication Developers Social Network
re developers of the collaboration Developers Social Network
e developers present only in the communication DSN
e developers present only in the collaboration DSN
e developers present both in the communication and in the collaboration DSNs
evelopers present only in the communication DSN
evelopers present only in the collaboration DSN
evelopers present both in the communication and in the collaboration DSNs
ers turnover with respect to the previous temporal window
levelopers turnover with respect to the previous temporal window
velopers turnover with respect to the previous temporal window
cation developers turnover with respect to the previous temporal window
ion developers turnover with respect to the previous temporal window
pers previously involved in any Community Smell that left the community
etric of the global DSN computed using closeness
etric of the global DSN computed using betweenness
etric of the global DSN computed using degree
ty metric of the global DSN
ty metric of the communication Developers Social Network
ty metric of the collaboration Developers Social Network
letric of the global Developers Social Network



Truck Factor: Also known as the *bus factor*, is a measure of the risk within a software development project that arises from knowledge concentration. It represents the **minimum number of developers who**, **if incapacitated** (e.g., "hit by a truck"), **would cause the project to face severe difficulties in being completed or maintained.**

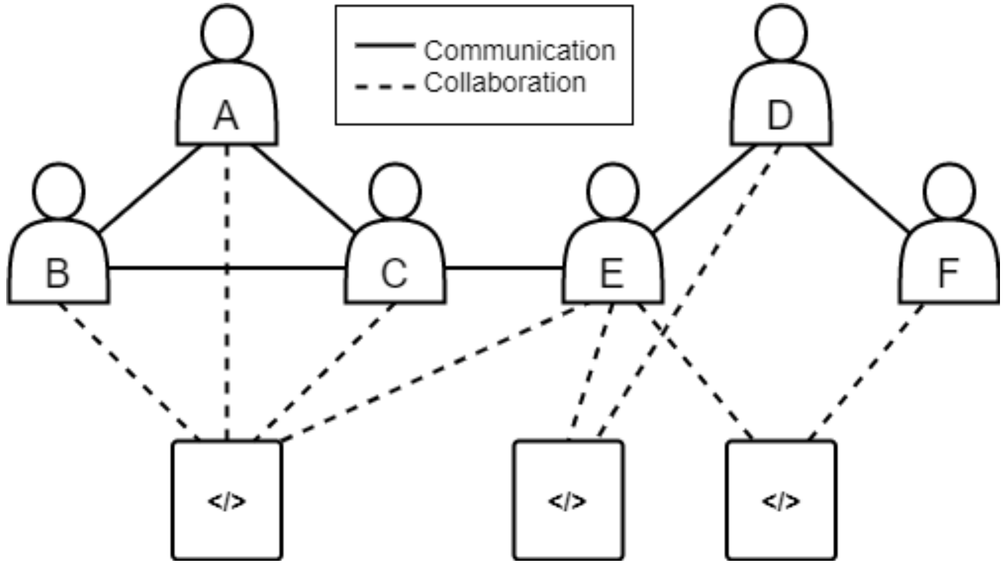
For example, if a project has a truck factor of 1, it means that only one developer holds critical knowledge or skills; if that developer were to leave, the project would be in jeopardy. Conversely, a higher truck factor indicates better knowledge distribution and a more resilient project structure.

elopers present in the global Developers Social Network
elopers present only in the communication Developers Social Network
elopers present only in the collaboration Developers Social Network
elopers present both in the collaboration and in the communication DSNs
levelopers present only in the communication Developers Social Network
levelopers present only in the collaboration Developers Social Network
levelopers present both in the collaboration and in the communication DSNs
nsored developers (95% of their commits are done in working hours)
ored developers with respect to developers present in the collaboration DSN
ocio-technical congruence
nformation communicability (decisions diffusion)
ezones involved in the software development
pers involved in at least one Community Smell
e developers of the global Developers Social Network
e developers of the communication Developers Social Network
e developers of the collaboration Developers Social Network
e sponsored developers
ponsored developers with respect to core developers of the collaboration DSN
re developers of the global Developers Social Network
re developers of the communication Developers Social Network
re developers of the collaboration Developers Social Network
e developers present only in the communication DSN
e developers present only in the collaboration DSN
e developers present both in the communication and in the collaboration DSNs
evelopers present only in the communication DSN
evelopers present only in the collaboration DSN
evelopers present both in the communication and in the collaboration DSNs
ers turnover with respect to the previous temporal window
levelopers turnover with respect to the previous temporal window
velopers turnover with respect to the previous temporal window
cation developers turnover with respect to the previous temporal window
ion developers turnover with respect to the previous temporal window
pers previously involved in any Community Smell that left the community
etric of the global DSN computed using closeness
etric of the global DSN computed using betweenness
etric of the global DSN computed using degree
y metric of the global DSN
y metric of the communication Developers Social Network
y metric of the collaboration Developers Social Network
letric of the global Developers Social Network



Socio-technical congruence: Alignment between **the social interactions within a development team** (the "social" aspect) and **the dependencies in the technical tasks they work on** (the "technical" aspect). This concept is crucial in collaborative software development, where teams must effectively communicate and coordinate according to the dependencies inherent in their tasks.

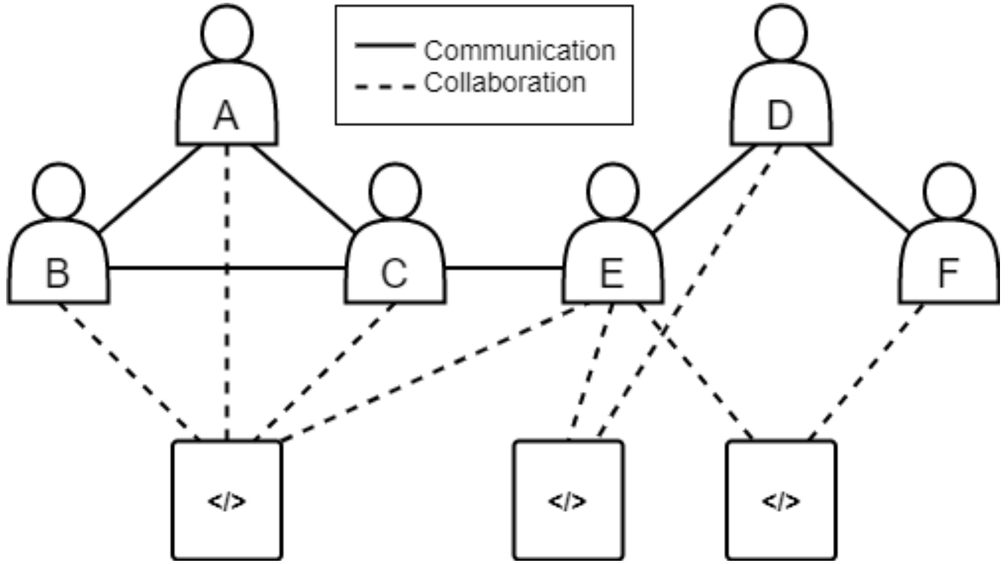
elopers present in the global Developers Social Network
elopers present only in the communication Developers Social Network
elopers present only in the collaboration Developers Social Network
elopers present both in the collaboration and in the communication DSNs
levelopers present only in the communication Developers Social Network
levelopers present only in the collaboration Developers Social Network
levelopers present both in the collaboration and in the communication DSNs
nsored developers (95% of their commits are done in working hours)
ored developers with respect to developers present in the collaboration DSN
ocio-technical congruence
nformation communicability (decisions diffusion)
ezones involved in the software development
pers involved in at least one Community Smell
e developers of the global Developers Social Network
e developers of the communication Developers Social Network
e developers of the collaboration Developers Social Network
e sponsored developers
ponsored developers with respect to core developers of the collaboration DSN
re developers of the global Developers Social Network
re developers of the communication Developers Social Network
re developers of the collaboration Developers Social Network
e developers present only in the communication DSN
e developers present only in the collaboration DSN
e developers present both in the communication and in the collaboration DSNs
evelopers present only in the communication DSN
evelopers present only in the collaboration DSN
evelopers present both in the communication and in the collaboration DSNs
ers turnover with respect to the previous temporal window
levelopers turnover with respect to the previous temporal window
velopers turnover with respect to the previous temporal window
cation developers turnover with respect to the previous temporal window
ion developers turnover with respect to the previous temporal window
pers previously involved in any Community Smell that left the community
etric of the global DSN computed using closeness
etric of the global DSN computed using betweenness
etric of the global DSN computed using degree
y metric of the global DSN
y metric of the communication Developers Social Network
y metric of the collaboration Developers Social Network
etric of the global Developers Social Network



Socio-technical congruence: Alignment between **the social interactions within a development team** (the "social" aspect) and **the dependencies in the technical tasks they work on** (the "technical" aspect). This concept is crucial in collaborative software development, where teams must effectively communicate and coordinate according to the dependencies inherent in their tasks.

Using a socio-technical graph, one can map the alignment between communication and collaboration activities, understanding how developers interact while developing artifacts.

elopers present in the global Developers Social Network
elopers present only in the communication Developers Social Network
elopers present only in the collaboration Developers Social Network
elopers present both in the collaboration and in the communication DSNs
levelopers present only in the communication Developers Social Network
levelopers present only in the collaboration Developers Social Network
levelopers present both in the collaboration and in the communication DSNs
nsored developers (95% of their commits are done in working hours)
ored developers with respect to developers present in the collaboration DSN
ocio-technical congruence
nformation communicability (decisions diffusion)
ezones involved in the software development
pers involved in at least one Community Smell
e developers of the global Developers Social Network
e developers of the communication Developers Social Network
e developers of the collaboration Developers Social Network
e sponsored developers
ponsored developers with respect to core developers of the collaboration DSN
re developers of the global Developers Social Network
re developers of the communication Developers Social Network
re developers of the collaboration Developers Social Network
e developers present only in the communication DSN
e developers present only in the collaboration DSN
e developers present both in the communication and in the collaboration DSNs
evelopers present only in the communication DSN
evelopers present only in the collaboration DSN
evelopers present both in the communication and in the collaboration DSNs
ers turnover with respect to the previous temporal window
levelopers turnover with respect to the previous temporal window
velopers turnover with respect to the previous temporal window
cation developers turnover with respect to the previous temporal window
ion developers turnover with respect to the previous temporal window
pers previously involved in any Community Smell that left the community
etric of the global DSN computed using closeness
etric of the global DSN computed using betweenness
etric of the global DSN computed using degree
y metric of the global DSN
y metric of the communication Developers Social Network
y metric of the collaboration Developers Social Network
letric of the global Developers Social Network



Socio-technical congruence: Alignment between **the social interactions within a development team** (the "social" aspect) and **the dependencies in the technical tasks they work on** (the "technical" aspect). This concept is crucial in collaborative software development, where teams must effectively communicate and coordinate according to the dependencies inherent in their tasks.

Using a socio-technical graph, one can map the alignment between communication and collaboration activities, understanding how developers interact while developing artifacts.

Why is socio-technical congruence so important?

elopers present in the global Developers Social Network
elopers present only in the communication Developers Social Network
elopers present only in the collaboration Developers Social Network
elopers present both in the collaboration and in the communication DSNs
levelopers present only in the communication Developers Social Network
levelopers present only in the collaboration Developers Social Network
levelopers present both in the collaboration and in the communication DSNs
nsored developers (95% of their commits are done in working hours)
ored developers with respect to developers present in the collaboration DSN

ocio-technical congruence
nformation communicability (decisions diffusion)
ezones involved in the software development
pers involved in at least one Community Smell

e developers of the global Developers Social Network
e developers of the communication Developers Social Network
e developers of the collaboration Developers Social Network
e sponsored developers
ponsored developers with respect to core developers of the collaboration DSN
re developers of the global Developers Social Network
re developers of the communication Developers Social Network
re developers of the collaboration Developers Social Network
e developers present only in the communication DSN
e developers present only in the collaboration DSN
e developers present both in the communication and in the collaboration DSNs
evelopers present only in the communication DSN
evelopers present only in the collaboration DSN
evelopers present both in the communication and in the collaboration DSNs

ers turnover with respect to the previous temporal window
levelopers turnover with respect to the previous temporal window
velopers turnover with respect to the previous temporal window
cation developers turnover with respect to the previous temporal window
ion developers turnover with respect to the previous temporal window
pers previously involved in any Community Smell that left the community

etric of the global DSN computed using closeness
etric of the global DSN computed using betweenness
etric of the global DSN computed using degree
ty metric of the global DSN
ty metric of the communication Developers Social Network
ty metric of the collaboration Developers Social Network
letric of the global Developers Social Network

Does Socio-Technical Congruence Have an Effect on Software Build Success? A Study of Coordination in a Software Project

Irwin Kwan, *Student Member*, *IEEE*, Adrian Schröter, *Student Member*, *IEEE*, and Daniela Damian, *Member*, *IEEE Computer Society*

Abstract—Socio-technical congruence is an approach that measures coordination by examining the alignment between the technical dependencies and the social coordination in the project. We conduct a case study of coordination in the IBM Rational Team Concert project, which consists of 151 developers over seven geographically distributed sites, and expect that high congruence leads to a high probability of successful builds. We examine this relationship by applying two congruence measurements: an unweighted congruence measure from previous literature, and a weighted measure that overcomes limitations of the existing measure. We discover that there is a relationship between socio-technical congruence and build success probability, but only for certain build types, and observe that in some situations, higher congruence actually leads to lower build success rates. We also observe that a large proportion of zero-congruence builds are successful, and that socio-technical gaps in successful builds are larger than gaps in failed builds. Analysis of the social and technical aspects in IBM Rational Team Concert allows us to discuss the effects of congruence on build success. Our findings provide implications with respect to the limits of applicability of socio-technical congruence and suggest further improvements of socio-technical congruence to study coordination.

Index Terms—Empirical software engineering, socio-technical congruence, coordination, awareness, software quality, integration.

1 Introduction

Coordinating the efforts of individuals working together in a team is necessary to build software systems. The complexity of current systems requires contributions from tens or hundreds of people who may span multiple offices, cities, or even continents. To build such systems, we need to ensure that the team is not only capable of developing components of a system, but also has the governance to be able to integrate the interdependent parts into a whole.

We describe a case study of socio-technical coordination and its effect on software builds in the IBM Rational Team Concert (RTC) software product.[1] Our approach to investigating coordination is to examine the alignment between the technical dimension of work and the social relationships between team members. This alignment is called **socio-technical congruence** [1]. High socio-technical congruence has been shown to be a predictor of coordination success [1], [2]. The mismatches between the social and technical dimensions, or **gaps**, also have been observed as increasing resolution times for software activities. The objective of this study is to investigate the effects of socio-technical congruence on high-coordination software development activities during a project.

We seek to discover the relationship that congruence has on the probability that a regularly scheduled software build will be successful. We conduct a case study of a large software project at IBM. A build result, which can be *error* or *OK*, indicates the relative health of the project up to that build. To measure socio-technical congruence we apply two different measures: a previously published congruence approach [1], and a weighted congruence approach that provides details about the size of a gap between two individuals [3]. We also examine RTC's processes and tools to identify any explanations of the relationship between congruence and builds that we find.

Our results indicate that congruence affects build success probabilities for certain types of builds. In a continuous build which may involve changes submitted primarily by a colocated team, high congruence leads to a higher probability of build success. However, in an integration build, which involves code submitted by many distributed teams, high congruence actually reduces the probability of build success. Although the congruence per build is relatively low at approximately 20 percent overall, the team is able to coordinate to complete their builds. We also find that the mean size of a congruence gap correlates inversely with build success probability. Our findings on the socio-technical congruence and build quality in RTC are complemented by insights we obtained by studying the context of the RTC project. Some of these results may be explained also by the congruence conceptualizations that we use in our empirical study.

1. IBM, Rational, Jazz and Rational Team Concert are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

• *The authors are with the Software Engineering Global interAction Lab, Department of Computer Science, University of Victoria, Victoria, PO Box 3055, STN CSC Victoria, BC V8W 3P6, Canada.*
E-mail: irwink@ieee.org, schadr@acm.org, danielad@cs.uvic.ca.

Manuscript received 15 Oct. 2009; revised 19 Apr. 2010; accepted 7 Dec. 2010; published online 18 Mar. 2011.
Recommended for acceptance by M. Cataldo and A. Mockus.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSESI-2009-10-0257.
Digital Object Identifier no. 10.1109/TSE.2011.29.

0098-5589/11/$26.00 © 2011 IEEE Published by the IEEE Computer Society

elopers present in the global Developers Social Network
elopers present only in the communication Developers Social Network
elopers present only in the collaboration Developers Social Network
elopers present both in the collaboration and in the communication DSNs
levelopers present only in the communication Developers Social Network
levelopers present only in the collaboration Developers Social Network
levelopers present both in the collaboration and in the communication DSNs
nsored developers (95% of their commits are done in working hours)
ored developers with respect to developers present in the collaboration DSN
ocio-technical congruence
nformation communicability (decisions diffusion)
ezones involved in the software development
pers involved in at least one Community Smell
e developers of the global Developers Social Network
e developers of the communication Developers Social Network
e developers of the collaboration Developers Social Network
e sponsored developers
ponsored developers with respect to core developers of the collaboration DSN
re developers of the global Developers Social Network
re developers of the communication Developers Social Network
re developers of the collaboration Developers Social Network
e developers present only in the communication DSN
e developers present only in the collaboration DSN
e developers present both in the communication and in the collaboration DSNs
evelopers present only in the communication DSN
evelopers present only in the collaboration DSN
evelopers present both in the communication and in the collaboration DSNs
ers turnover with respect to the previous temporal window
levelopers turnover with respect to the previous temporal window
velopers turnover with respect to the previous temporal window
cation developers turnover with respect to the previous temporal window
ion developers turnover with respect to the previous temporal window
pers previously involved in any Community Smell that left the community
etric of the global DSN computed using closeness
etric of the global DSN computed using betweenness
etric of the global DSN computed using degree
ty metric of the global DSN
ty metric of the communication Developers Social Network
ty metric of the collaboration Developers Social Network
letric of the global Developers Social Network

When developers working on interdependent components communicate effectively, they're more likely to share crucial details, such as interface changes, function modifications, or potential integration challenges. Without this communication, dependencies may break, leading to failed builds.

Does Socio-Technical Congruence Have an Effect on Software Build Success? A Study of Coordination in a Software Project

Irwin Kwan, *Student Member, IEEE*, Adrian Schröter, *Student Member, IEEE*, and Daniela Damian, *Member, IEEE Computer Society*

Abstract—Socio-technical congruence is an approach that measures coordination by examining the alignment between the technical dependencies and the social coordination in the project. We conduct a case study of coordination in the IBM Rational Team Concert project, which consists of 151 developers over seven geographically distributed sites, and expect that high congruence leads to a high probability of successful builds. We examine this relationship by applying two congruence measurements: an unweighted congruence measure from previous literature, and a weighted measure that overcomes limitations of the existing measure. We discover that there is a relationship between socio-technical congruence and build success probability, but only for certain build types, and observe that in some situations, higher congruence actually leads to lower build success rates. We also observe that a large proportion of zero-congruence builds are successful, and that socio-technical gaps in successful builds are larger than gaps in failed builds. Analysis of the social and technical aspects in IBM Rational Team Concert allows us to discuss the effects of congruence on build success. Our findings provide implications with respect to the limits of applicability of socio-technical congruence and suggest further improvements

congruence, coordination, awareness, software quality, integration.

able to integrate the interdependent parts into a whole.

We describe a case study of socio-technical coordination and its effect on software builds in the IBM Rational Team Concert (RTC) software product.[1] Our approach to investigating coordination is to examine the alignment between the technical dimension of work and the social relationships between team members. This alignment is called **socio-technical congruence** [1]. High socio-technical congruence has been shown to be a predictor of coordination success [1], [2]. The mismatches between the social and technical dimensions, or **gaps**, also have been observed as increasing resolution times for software activities. The objective of this study is to investigate the effects of socio-technical congruence on high-coordination software development activities during a project.

We seek to discover the relationship that congruence has on the probability that a regularly scheduled software build will be successful. We conduct a case study of a large software project at IBM. A build result, which can be *error* or *OK*, indicates the relative health of the project up to that build. To measure socio-technical congruence we apply two different measures: a previously published congruence approach [1], and a weighted congruence approach that provides details about the size of a gap between two individuals [3]. We also examine RTC's processes and tools to identify any explanations of the relationship between congruence and builds that we find.

Our results indicate that congruence affects build success probabilities for certain types of builds. In a continuous build which may involve changes submitted primarily by a colocated team, high congruence leads to a higher probability of build success. However, in an integration build, which involves code submitted by many distributed teams, high congruence actually reduces the probability of build success. Although the congruence per build is relatively low at approximately 20 percent overall, the team is able to coordinate to complete their builds. We also find that the mean size of a congruence gap correlates inversely with build success probability. Our findings on the socio-technical congruence and build quality in RTC are complemented by insights we obtained by studying the context of the RTC project. Some of these results may be explained also by the congruence conceptualizations that we use in our empirical study.

1. IBM, Rational, Jazz and Rational Team Concert are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

• *The authors are with the Software Engineering Global interAction Lab, Department of Computer Science, University of Victoria, Victoria, PO Box 3055, STN CSC Victoria, BC V8W 3P6, Canada.*
E-mail: irwink@ieee.org, schadr@acm.org, danielad@cs.uvic.ca.

Manuscript received 15 Oct. 2009; revised 19 Apr. 2010; accepted 7 Dec. 2010; published online 18 Mar. 2011.
Recommended for acceptance by M. Cataldo and A. Mockus.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSESI-2009-10-0257.
Digital Object Identifier no. 10.1109/TSE.2011.29.

0098-5589/11/$26.00 © 2011 IEEE Published by the IEEE Computer Society

elopers present in the global Developers Social Network
elopers present only in the communication Developers Social Network
elopers present only in the collaboration Developers Social Network
elopers present both in the collaboration and in the communication DSNs
levelopers present only in the communication Developers Social Network
levelopers present only in the collaboration Developers Social Network
levelopers present both in the collaboration and in the communication DSNs
nsored developers (95% of their commits are done in working hours)
ored developers with respect to developers present in the collaboration DSN
ocio-technical congruence
nformation communicability (decisions diffusion)
ezones involved in the software development
pers involved in at least one Community Smell
e developers of the global Developers Social Network
e developers of the communication Developers Social Network
e developers of the collaboration Developers Social Network
e sponsored developers
ponsored developers with respect to core developers of the collaboration DSN
re developers of the global Developers Social Network
re developers of the communication Developers Social Network
re developers of the collaboration Developers Social Network
e developers present only in the communication DSN
e developers present only in the collaboration DSN
e developers present both in the communication and in the collaboration DSNs
evelopers present only in the communication DSN
evelopers present only in the collaboration DSN
evelopers present both in the communication and in the collaboration DSNs
ers turnover with respect to the previous temporal window
levelopers turnover with respect to the previous temporal window
velopers turnover with respect to the previous temporal window
cation developers turnover with respect to the previous temporal window
ion developers turnover with respect to the previous temporal window
pers previously involved in any Community Smell that left the community
etric of the global DSN computed using closeness
etric of the global DSN computed using betweenness
etric of the global DSN computed using degree
ty metric of the global DSN
ty metric of the communication Developers Social Network
ty metric of the collaboration Developers Social Network
letric of the global Developers Social Network

When developers working on interdependent components communicate effectively, they're more likely to share crucial details, such as interface changes, function modifications, or potential integration challenges. Without this communication, dependencies may break, leading to failed builds.

High congruence aligns teams on coding standards, testing practices, and other protocols, reducing the likelihood of errors that disrupt build success. Misalignment can lead to inconsistent practices, resulting in bugs or incompatible code contributions.

Does Socio-Technical Congruence Have an Effect on Software Build Success? A Study of Coordination in a Software Project

Irwin Kwan, *Student Member, IEEE*, Adrian Schröter, *Student Member, IEEE*, and Daniela Damian, *Member, IEEE Computer Society*

Abstract—Socio-technical congruence is an approach that measures coordination by examining the alignment between the technical dependencies and the social coordination in the project. We conduct a case study of coordination in the IBM Rational Team Concert project, which consists of 151 developers over seven geographically distributed sites, and expect that high congruence leads to a high probability of successful builds. We examine this relationship by applying two congruence measurements: an unweighted congruence measure from previous literature, and a weighted measure that overcomes limitations of the existing measure. We discover that there is a relationship between socio-technical congruence and build success probability, but only for certain build types, and observe that in some situations, higher congruence actually leads to lower build success rates. We also observe that a large proportion of zero-congruence builds are successful, and that socio-technical gaps in successful builds are larger than gaps in failed builds. Analysis of the social and technical aspects in IBM Rational Team Concert allows us to discuss the effects of congruence on build success. Our findings provide implications with respect to the limits of applicability of socio-technical congruence and suggest further improvements

congruence, coordination, awareness, software quality, integration.

✦

able to integrate the interdependent parts into a whole.

activities. The objective of this study is to investigate the effects of socio-technical congruence on high-coordination software development activities during a project.

We seek to discover the relationship that congruence has on the probability that a regularly scheduled software build will be successful. We conduct a case study of a large software project at IBM. A build result, which can be *error* or *OK*, indicates the relative health of the project up to that build. To measure socio-technical congruence we apply two different measures: a previously published congruence approach [1], and a weighted congruence approach that provides details about the size of a gap between two individuals [3]. We also examine RTC's processes and tools to identify any explanations of the relationship between congruence and builds that we find.

Our results indicate that congruence affects build success probabilities for certain types of builds. In a continuous build which may involve changes submitted primarily by a colocated team, high congruence leads to a higher probability of build success. However, in an integration build, which involves code submitted by many distributed teams, high congruence actually reduces the probability of build success. Although the congruence per build is relatively low at approximately 20 percent overall, the team is able to coordinate to complete their builds. We also find that the mean size of a congruence gap correlates inversely with build success probability. Our findings on the socio-technical congruence and build quality in RTC are complemented by insights we obtained by studying the context of the RTC project. Some of these results may be explained also by the congruence conceptualizations that we use in our empirical study.

• *The authors are with the Software Engineering Global interAction Lab, Department of Computer Science, University of Victoria, Victoria, PO Box 3055, STN CSC Victoria, BC V8W 3P6, Canada.*
E-mail: irwink@ieee.org, schadr@acm.org, danielad@cs.uvic.ca.

Manuscript received 15 Oct. 2009; revised 19 Apr. 2010; accepted 7 Dec. 2010; published online 18 Mar. 2011.
Recommended for acceptance by M. Cataldo and A. Mockus.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSESI-2009-10-0257.
Digital Object Identifier no. 10.1109/TSE.2011.29.

0098-5589/11/$26.00 © 2011 IEEE Published by the IEEE Computer Society

elopers present in the global Developers Social Network
elopers present only in the communication Developers Social Network
elopers present only in the collaboration Developers Social Network
elopers present both in the collaboration and in the communication DSNs
levelopers present only in the communication Developers Social Network
levelopers present only in the collaboration Developers Social Network
levelopers present both in the collaboration and in the communication DSNs
nsored developers (95% of their commits are done in working hours)
ored developers with respect to developers present in the collaboration DSN
ocio-technical congruence
nformation communicability (decisions diffusion)
ezones involved in the software development
pers involved in at least one Community Smell
e developers of the global Developers Social Network
e developers of the communication Developers Social Network
e developers of the collaboration Developers Social Network
e sponsored developers
ponsored developers with respect to core developers of the collaboration DSN
re developers of the global Developers Social Network
re developers of the communication Developers Social Network
re developers of the collaboration Developers Social Network
e developers present only in the communication DSN
e developers present only in the collaboration DSN
e developers present both in the communication and in the collaboration DSNs
evelopers present only in the communication DSN
evelopers present only in the collaboration DSN
evelopers present both in the communication and in the collaboration DSNs
ers turnover with respect to the previous temporal window
levelopers turnover with respect to the previous temporal window
velopers turnover with respect to the previous temporal window
cation developers turnover with respect to the previous temporal window
ion developers turnover with respect to the previous temporal window
pers previously involved in any Community Smell that left the community
etric of the global DSN computed using closeness
etric of the global DSN computed using betweenness
etric of the global DSN computed using degree
ty metric of the global DSN
ty metric of the communication Developers Social Network
ty metric of the collaboration Developers Social Network
letric of the global Developers Social Network

When developers working on interdependent components communicate effectively, they're more likely to share crucial details, such as interface changes, function modifications, or potential integration challenges. Without this communication, dependencies may break, leading to failed builds.

High congruence aligns teams on coding standards, testing practices, and other protocols, reducing the likelihood of errors that disrupt build success. Misalignment can lead to inconsistent practices, resulting in bugs or incompatible code contributions.

In complex projects, changes in one component can ripple across other parts of the system. Socio-technical congruence helps ensure that these changes are communicated across the relevant teams, allowing everyone affected to make necessary adjustments before integration, thus reducing build failures.

Does Socio-Technical Congruence Have an Effect on Software Build Success? A Study of Coordination in a Software Project

Irwin Kwan, *Student Member, IEEE*, Adrian Schröter, *Student Member, IEEE*, and Daniela Damian, *Member, IEEE Computer Society*

Abstract—Socio-technical congruence is an approach that measures coordination by examining the alignment between the technical dependencies and the social coordination in the project. We conduct a case study of coordination in the IBM Rational Team Concert project, which consists of 151 developers over seven geographically distributed sites, and expect that high congruence leads to a high probability of successful builds. We examine this relationship by applying two congruence measurements: an unweighted congruence measure from previous literature, and a weighted measure that overcomes limitations of the existing measure. We discover that there is a relationship between socio-technical congruence and build success probability, but only for certain build types, and observe that in some situations, higher congruence actually leads to lower build success rates. We also observe that a large proportion of zero-congruence builds are successful, and that socio-technical gaps in successful builds are larger than gaps in failed builds. Analysis of the social and technical aspects in IBM Rational Team Concert allows us to discuss the effects of congruence on build success. Our findings provide implications with respect to the limits of applicability of socio-technical congruence and suggest further improvements

congruence, coordination, awareness, software quality, integration.

able to integrate the interdependent parts into a whole.

activities. The objective of this study is to investigate the effects of socio-technical congruence on high-coordination software development activities during a project.

We seek to discover the relationship that congruence has on the probability that a regularly scheduled software build will be successful. We conduct a case study of a large software project at IBM. A build result, which can be *error* or *OK*, indicates the relative health of the project up to that build. To measure socio-technical congruence we apply two different measures: a previously published congruence approach [1], and a weighted congruence approach that provides details about the size of a gap between two individuals [3]. We also examine RTC's processes and tools to identify any explanations of the relationship between congruence and builds that we find.

Our results indicate that congruence affects build success probabilities for certain types of builds. In a continuous build which may involve changes submitted primarily by a colocated team, high congruence leads to a higher probability of build success. However, in an integration build, which involves code submitted by many distributed teams, high congruence actually reduces the probability of build success. Although the congruence per build is relatively low at approximately 20 percent overall, the team is able to coordinate to complete their builds. We also find that the mean size of a congruence gap correlates inversely with build success probability. Our findings on the socio-technical congruence and build quality in RTC are complemented by insights we obtained by studying the context of the RTC project. Some of these results may be explained also by the congruence conceptualizations that we use in our empirical study.

elopers present in the global Developers Social Network
elopers present only in the communication Developers Social Network
elopers present only in the collaboration Developers Social Network
elopers present both in the collaboration and in the communication DSNs
levelopers present only in the communication Developers Social Network
levelopers present only in the collaboration Developers Social Network
levelopers present both in the collaboration and in the communication DSNs
nsored developers (95% of their commits are done in working hours)
ored developers with respect to developers present in the collaboration DSN
ocio-technical congruence
nformation communicability (decisions diffusion)
ezones involved in the software development
pers involved in at least one Community Smell
e developers of the global Developers Social Network
e developers of the communication Developers Social Network
e developers of the collaboration Developers Social Network
e sponsored developers
ponsored developers with respect to core developers of the collaboration DSN
re developers of the global Developers Social Network
re developers of the communication Developers Social Network
re developers of the collaboration Developers Social Network
e developers present only in the communication DSN
e developers present only in the collaboration DSN
e developers present both in the communication and in the collaboration DSNs
evelopers present only in the communication DSN
evelopers present only in the collaboration DSN
evelopers present both in the communication and in the collaboration DSNs
ers turnover with respect to the previous temporal window
levelopers turnover with respect to the previous temporal window
velopers turnover with respect to the previous temporal window
cation developers turnover with respect to the previous temporal window
ion developers turnover with respect to the previous temporal window
pers previously involved in any Community Smell that left the community
etric of the global DSN computed using closeness
etric of the global DSN computed using betweenness
etric of the global DSN computed using degree
ty metric of the global DSN
ty metric of the communication Developers Social Network
ty metric of the collaboration Developers Social Network
letric of the global Developers Social Network

Socio-Technical Congruence: A Framework for Assessing the Impact of Technical and Work Dependencies on Software Development Productivity

Marcelo Cataldo
Research and Technology Center
Bosch Corporate Research
Pittsburgh, PA 15212, USA
marcelo.cataldo@us.bosch.com

James D. Herbsleb
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
jdh@cs.cmu.edu

Kathleen M. Carley
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
Kathleen.carley@cmu.edu

ABSTRACT

The identification and management of work dependencies is a fundamental challenge in software development organizations. This paper argues that modularization, the traditional technique intended to reduce interdependencies among components of a system, has serious limitations in the context of software development. We build on the idea of congruence, proposed in our prior work, to examine the relationship between the structure of technical and work dependencies and the impact of dependencies on software development productivity. Our empirical evaluation of the congruence framework showed that when developers' coordination patterns are congruent with their coordination needs, the resolution time of modification requests was significantly reduced. Furthermore, our analysis highlights the importance of identifying the "right" set of technical dependencies that drive the coordination requirements among software developers. Call and data dependencies appear to have far less impact than logical dependencies.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management – *Productivity, Programming Teams.* K.6.1 [**Management of computing and information Systems**]: Project and People Management – *Software development.*

General Terms

Management, Measurement, Human Factors.

Keywords

Collaborative software development, coordination, software dependencies.

ESEM'08, October 9–10, 2008, Kaiserslautern, Germany.

1. INTRODUCTION

A growing body of research shows that work dependencies – i.e., engineering decisions constraining other engineering decisions – is a fundamental challenge in software development organizations, particularly in those that are geographically distributed (e.g., [11][16][25][28]). The modular product design literature has extensively examined issues associated with dependencies. Design structure matrices, for example, have been used to find alternative structures that reduce dependencies among the components of a system [19][43]. These research streams can also inform the design of development organizations so they are better able to identify and manage work dependencies. However, we first need to understand the assumptions of the different theoretical views and how those assumptions relate to the characteristics of software development tasks.

This study argues that modularization is a necessary but not a sufficient mechanism for handling the work dependencies that emerge in the process of developing software systems. We build on the concept of congruence introduced by Cataldo et al [10] to examine how different types of technical dependencies relate to work dependencies among software developers and, ultimately, how those work dependencies impact development productivity. Our empirical evaluation of the congruence framework illustrates the importance of understanding the dynamic nature of software development. Identifying the "right" set of technical dependencies that determine the relevant work dependencies and coordinating accordingly has significant impact on reducing the resolution time of software modification requests. The analyses showed traditional software dependencies, such as syntactic relationships, tend to capture a relatively narrow view of product dependencies that is not fully representative of the important product dependencies that drive the need to coordinate. On the other hand, logical dependencies provide a more accurate representation of the product dependencies affecting the development effort.

The rest of this paper is organized as follows. We first discuss the theoretical background concerning the relationship between technical and work dependencies in software development projects. Next, we present the socio-technical congruence framework followed by a description of data, measures and models used in the empirical analysis. Finally, we discuss the results, their implications and future work.

elopers present in the global Developers Social Network
elopers present only in the communication Developers Social Network
elopers present only in the collaboration Developers Social Network
elopers present both in the collaboration and in the communication DSNs
levelopers present only in the communication Developers Social Network
levelopers present only in the collaboration Developers Social Network
levelopers present both in the collaboration and in the communication DSNs
nsored developers (95% of their commits are done in working hours)
ored developers with respect to developers present in the collaboration DSN
ocio-technical congruence
nformation communicability (decisions diffusion)
ezones involved in the software development
pers involved in at least one Community Smell
e developers of the global Developers Social Network
e developers of the communication Developers Social Network
e developers of the collaboration Developers Social Network
e sponsored developers
ponsored developers with respect to core developers of the collaboration DSN
re developers of the global Developers Social Network
re developers of the communication Developers Social Network
re developers of the collaboration Developers Social Network
e developers present only in the communication DSN
e developers present only in the collaboration DSN
e developers present both in the communication and in the collaboration DSNs
evelopers present only in the communication DSN
evelopers present only in the collaboration DSN
evelopers present both in the communication and in the collaboration DSNs
ers turnover with respect to the previous temporal window
levelopers turnover with respect to the previous temporal window
velopers turnover with respect to the previous temporal window
cation developers turnover with respect to the previous temporal window
ion developers turnover with respect to the previous temporal window
pers previously involved in any Community Smell that left the community
etric of the global DSN computed using closeness
etric of the global DSN computed using betweenness
etric of the global DSN computed using degree
ty metric of the global DSN
ty metric of the communication Developers Social Network
ty metric of the collaboration Developers Social Network
letric of the global Developers Social Network

When developers' coordination patterns are congruent with their coordination needs, the resolution time of modification requests was significantly reduced

Socio-Technical Congruence: A Framework for Assessing the Impact of Technical and Work Dependencies on Software Development Productivity

Marcelo Cataldo
Research and Technology Center
Bosch Corporate Research
Pittsburgh, PA 15212, USA
marcelo.cataldo@us.bosch.com

James D. Herbsleb
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
jdh@cs.cmu.edu

Kathleen M. Carley
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
Kathleen.carley@cmu.edu

ABSTRACT

The identification and management of work dependencies is a fundamental challenge in software development organizations. This paper argues that modularization, the traditional technique intended to reduce interdependencies among components of a system, has serious limitations in the context of software development. We build on the idea of congruence, proposed in our prior work, to examine the relationship between the structure of technical and work dependencies and the impact of dependencies on software development productivity. Our empirical evaluation of the congruence framework showed that when developers' coordination patterns are congruent with their coordination needs, the resolution time of modification requests was significantly reduced. Furthermore, our analysis highlights the importance of identifying the "right" set of technical dependencies that drive the coordination requirements among software developers. Call and data dependencies appear to have far less impact than logical dependencies.

ESEM'08, October 9–10, 2008, Kaiserslautern, Germany.

1. INTRODUCTION

A growing body of research shows that work dependencies – i.e., engineering decisions constraining other engineering decisions – is a fundamental challenge in software development organizations, particularly in those that are geographically distributed (e.g., [11][16][25][28]). The modular product design literature has extensively examined issues associated with dependencies. Design structure matrices, for example, have been used to find alternative structures that reduce dependencies among the components of a system [19][43]. These research streams can also inform the design of development organizations so they are better able to identify and manage work dependencies. However, we first need to understand the assumptions of the different theoretical views and how those assumptions relate to the characteristics of software development tasks.

This study argues that modularization is a necessary but not a sufficient mechanism for handling the work dependencies that emerge in the process of developing software systems. We build on the concept of congruence introduced by Cataldo et al [10] to examine how different types of technical dependencies relate to work dependencies among software developers and, ultimately, how those work dependencies impact development productivity. Our empirical evaluation of the congruence framework illustrates the importance of understanding the dynamic nature of software development. Identifying the "right" set of technical dependencies that determine the relevant work dependencies and coordinating accordingly has significant impact on reducing the resolution time of software modification requests. The analyses showed traditional software dependencies, such as syntactic relationships, tend to capture a relatively narrow view of product dependencies that is not fully representative of the important product dependencies that drive the need to coordinate. On the other hand, logical dependencies provide a more accurate representation of the product dependencies affecting the development effort.

The rest of this paper is organized as follows. We first discuss the theoretical background concerning the relationship between technical and work dependencies in software development projects. Next, we present the socio-technical congruence framework followed by a description of data, measures and models used in the empirical analysis. Finally, we discuss the results, their implications and future work.

elopers present in the global Developers Social Network
elopers present only in the communication Developers Social Network
elopers present only in the collaboration Developers Social Network
elopers present both in the collaboration and in the communication DSNs
levelopers present only in the communication Developers Social Network
levelopers present only in the collaboration Developers Social Network
levelopers present both in the collaboration and in the communication DSNs
nsored developers (95% of their commits are done in working hours)
ored developers with respect to developers present in the collaboration DSN
ocio-technical congruence
nformation communicability (decisions diffusion)
ezones involved in the software development
pers involved in at least one Community Smell
e developers of the global Developers Social Network
e developers of the communication Developers Social Network
e developers of the collaboration Developers Social Network
e sponsored developers
ponsored developers with respect to core developers of the collaboration DSN
re developers of the global Developers Social Network
re developers of the communication Developers Social Network
re developers of the collaboration Developers Social Network
e developers present only in the communication DSN
e developers present only in the collaboration DSN
e developers present both in the communication and in the collaboration DSNs
evelopers present only in the communication DSN
evelopers present only in the collaboration DSN
evelopers present both in the communication and in the collaboration DSNs
ers turnover with respect to the previous temporal window
levelopers turnover with respect to the previous temporal window
velopers turnover with respect to the previous temporal window
cation developers turnover with respect to the previous temporal window
ion developers turnover with respect to the previous temporal window
pers previously involved in any Community Smell that left the community
etric of the global DSN computed using closeness
etric of the global DSN computed using betweenness
etric of the global DSN computed using degree
ty metric of the global DSN
ty metric of the communication Developers Social Network
ty metric of the collaboration Developers Social Network
letric of the global Developers Social Network

When developers' coordination patterns are congruent with their coordination needs, the resolution time of modification requests was significantly reduced

While this is relevant in general, technologically-advanced systems, e.g., ML-enabled systems, are even largely impacted by severe issues if coordination is not congruent

Socio-Technical Congruence: A Framework for Assessing the Impact of Technical and Work Dependencies on Software Development Productivity

Marcelo Cataldo
Research and Technology Center
Bosch Corporate Research
Pittsburgh, PA 15212, USA
marcelo.cataldo@us.bosch.com

James D. Herbsleb
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
jdh@cs.cmu.edu

Kathleen M. Carley
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
Kathleen.carley@cmu.edu

ABSTRACT

The identification and management of work dependencies is a fundamental challenge in software development organizations. This paper argues that modularization, the traditional technique intended to reduce interdependencies among components of a system, has serious limitations in the context of software development. We build on the idea of congruence, proposed in our prior work, to examine the relationship between the structure of technical and work dependencies and the impact of dependencies on software development productivity. Our empirical evaluation of the congruence framework showed that when developers' coordination patterns are congruent with their coordination needs, the resolution time of modification requests was significantly reduced. Furthermore, our analysis highlights the importance of identifying the "right" set of technical dependencies that drive the coordination requirements among software developers. Call and data dependencies appear to have far less impact than logical dependencies.

1. INTRODUCTION

A growing body of research shows that work dependencies – i.e., engineering decisions constraining other engineering decisions – is a fundamental challenge in software development organizations, particularly in those that are geographically distributed (e.g., [11][16][25][28]). The modular product design literature has extensively examined issues associated with dependencies. Design structure matrices, for example, have been used to find alternative structures that reduce dependencies among the components of a system [19][43]. These research streams can also inform the design of development organizations so they are better able to identify and manage work dependencies. However, we first need to understand the assumptions of the different theoretical views and how those assumptions relate to the characteristics of software development tasks.

This study argues that modularization is a necessary but not a sufficient mechanism for handling the work dependencies that emerge in the process of developing software systems. We build on the concept of congruence introduced by Cataldo et al [10] to examine how different types of technical dependencies relate to work dependencies among software developers and, ultimately, how those work dependencies impact development productivity. Our empirical evaluation of the congruence framework illustrates the importance of understanding the dynamic nature of software development. Identifying the "right" set of technical dependencies that determine the relevant work dependencies and coordinating accordingly has significant impact on reducing the resolution time of software modification requests. The analyses showed traditional software dependencies, such as syntactic relationships, tend to capture a relatively narrow view of product dependencies that is not fully representative of the important product dependencies that drive the need to coordinate. On the other hand, logical dependencies provide a more accurate representation of the product dependencies affecting the development effort.

The rest of this paper is organized as follows. We first discuss the theoretical background concerning the relationship between technical and work dependencies in software development projects. Next, we present the socio-technical congruence framework followed by a description of data, measures and models used in the empirical analysis. Finally, we discuss the results, their implications and future work.

Software Project Management, How to measure?

Community Patterns

Social Debt

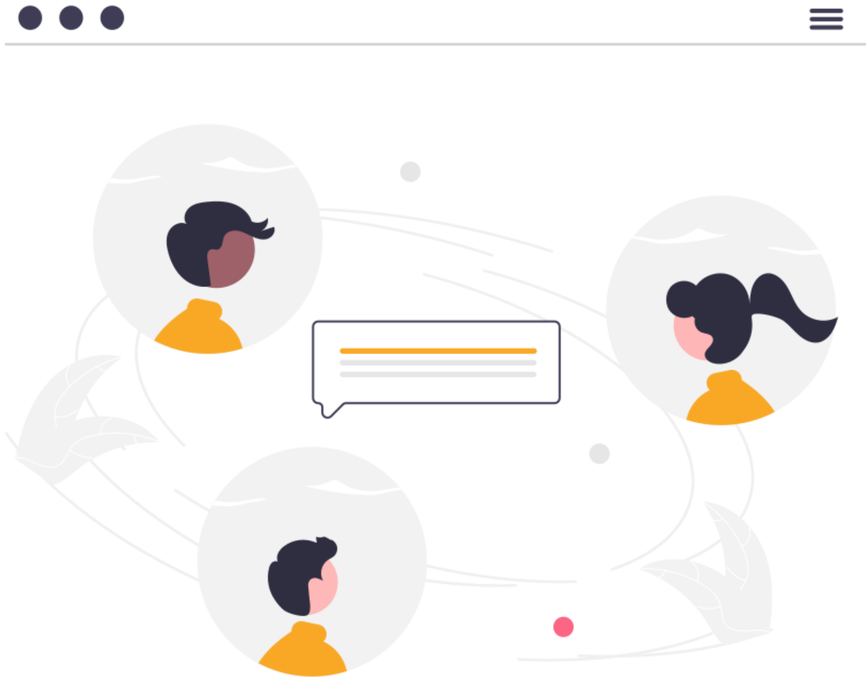
Software Project Management, How to measure?

Community Patterns

Social Debt

COMMUNITY

“An interacting population of various kinds of individuals (such as species) in a common location”



D. A. Tamburri, et al.
“Discovering community patterns in open-source:
a systematic approach and its evaluation.”
Empirical Software Engineering 24.3 (2019): 1369-1417.

YOSHI

Yielding Open-Source Health Information

TOAD

Towards Open-source communities health Analysis and Diagnosis

Tools capable of identifying the community patterns in an open-source community given a set of information and metrics that are obtainable mining a software repository

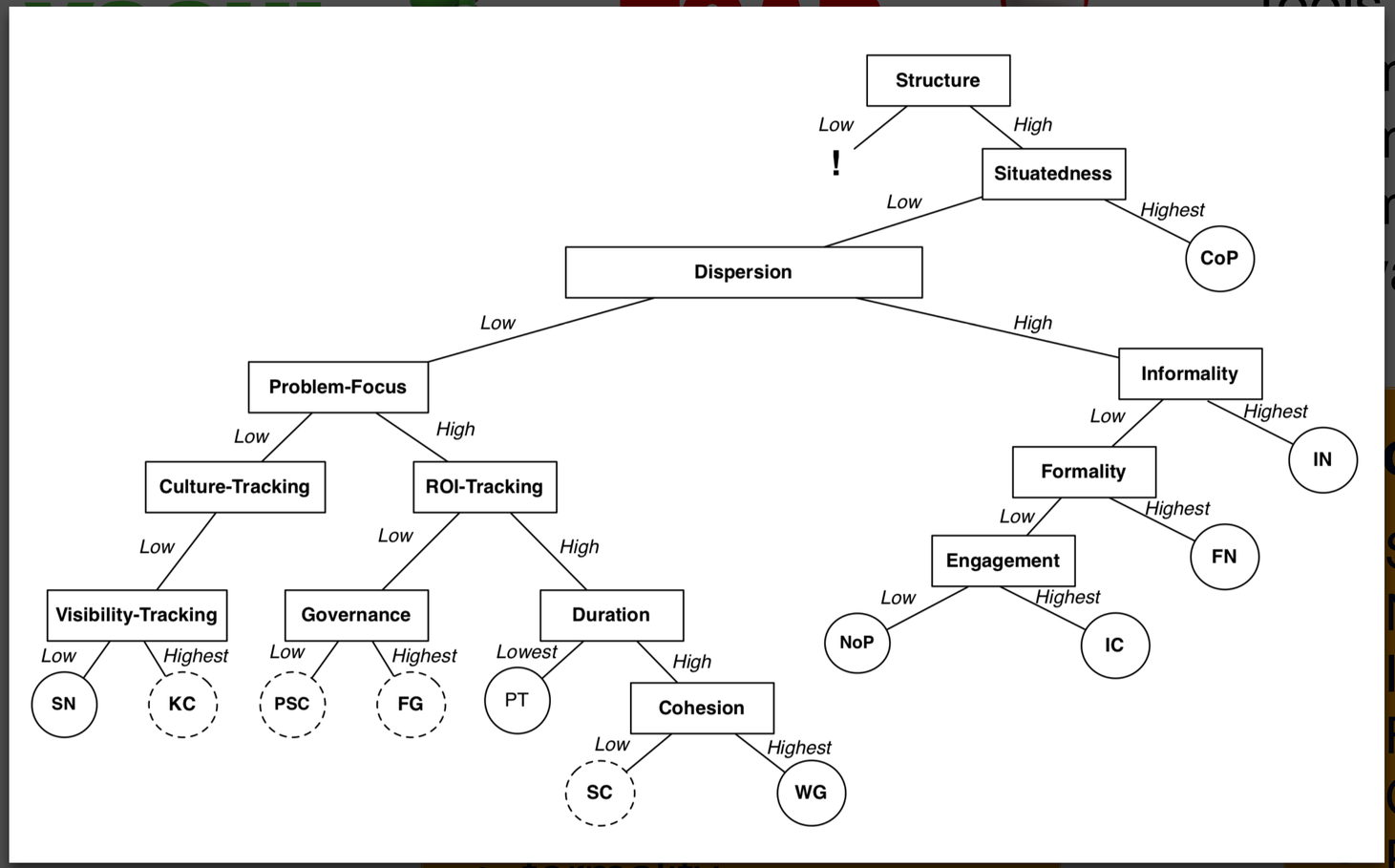
Measurement

- community structure
- geo-dispersion
- longevity
- engagement
- formality
- cohesion

Community Patterns

- Social Network (SN)
- Network of Practice (NoP)
- Informal Network (IN)
- Formal Network (FN)
- Community of Practice (CoP)
- Project Team (PT)
- Formal Group (FG)
- Informal Community (IC)

D. A. Tamburri, et al. “Discovering community patterns in open-source: a systematic approach and its evaluation.” Empirical Software Engineering 24.3 (2019): 1369-1417.



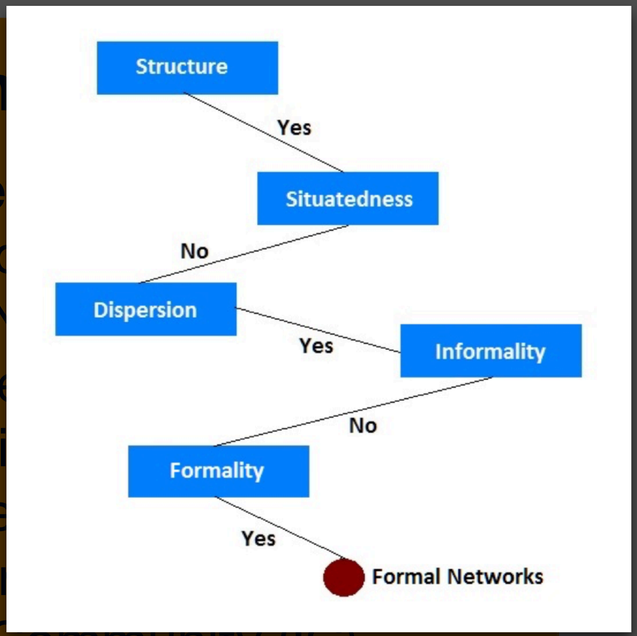
Tools capable of identifying the
nunity patterns in an open-source
nunity given a set of information
netrics that are obtainable mining a
are repository

- formality
- cohesion

ommun

Social Ne
Network
Informal N
Formal Ne
Communi
Project Te

- Formal Gr
- Informal Community (IC)



D. A. Tamburri, et al. “Discovering community patterns in open-source: a systematic approach and its evaluation.” Empirical Software Engineering 24.3 (2019): 1369-1417.

Software Project Management, How to measure?

Community Patterns

Social Debt

SOCIAL DEBT

Dove le Scienze Sociali incontrano l’Informatica per portare il caos

IMMAGINA!

Scenario di esempio

Avete passato la vostra intera vita a studiare, lavorare sodo e a scalare la gerarchia aziendale.

Dopo anni di sacrifici, finalmente il vostro capo vi ha promossi.

Ora siete dei MANAGER!

IMMAGINA!

Scenario di esempio

… nonostante ciò, gestire degli esseri umani può essere difficile (soprattutto se si tratta di ingegneri).

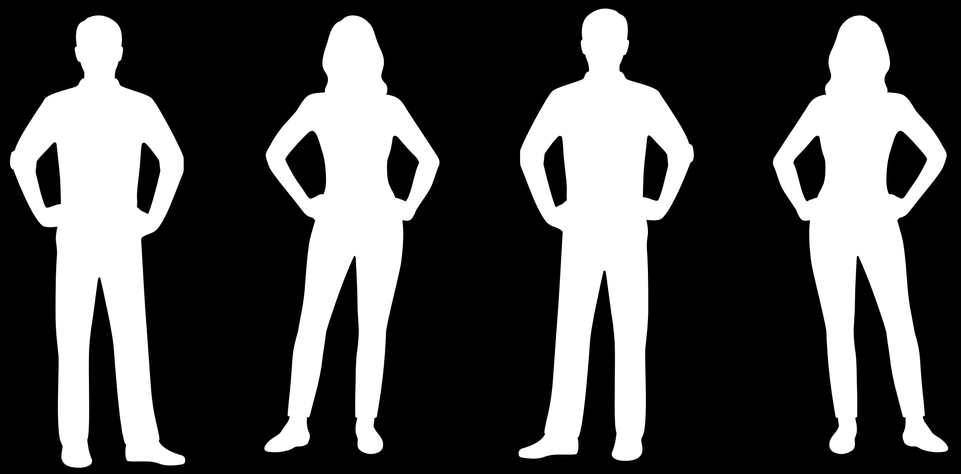
Prendere tale attività alla leggere può portare all’emergere di problemi (sociali) imprevisti e spesso distruttivi.


Merge che rompe tutto
Ritardi nel progetto
Malcontento del Team
Incremento dei costi


SOCIAL
DEBT

SOCIAL DEBT

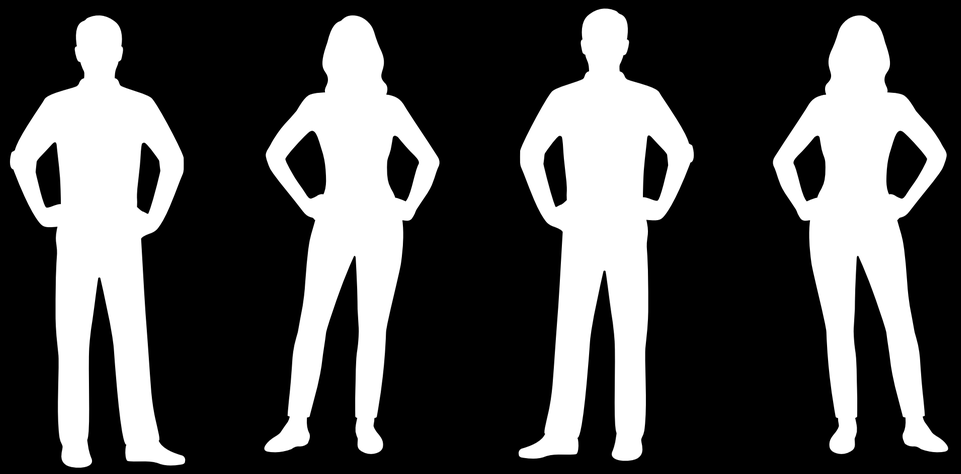
Sono dei costi aggiuntivi di un progetto software che derivano da ***socio-technical anti-pattern*** non previsti e non gestiti all'interno della comunità software [1].



[1] Tamburri, Damian A., et al. "What is social debt in software engineering?." 2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE). IEEE, 2013.

SOCIAL DEBT

Sono dei costi aggiuntivi di un progetto software che derivano da ***socio-technical anti-pattern*** non previsti e non gestiti all'interno della comunità software [1].



[1] Tamburri, Damian A., et al. "What is social debt in software engineering?." 2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE). IEEE, 2013.

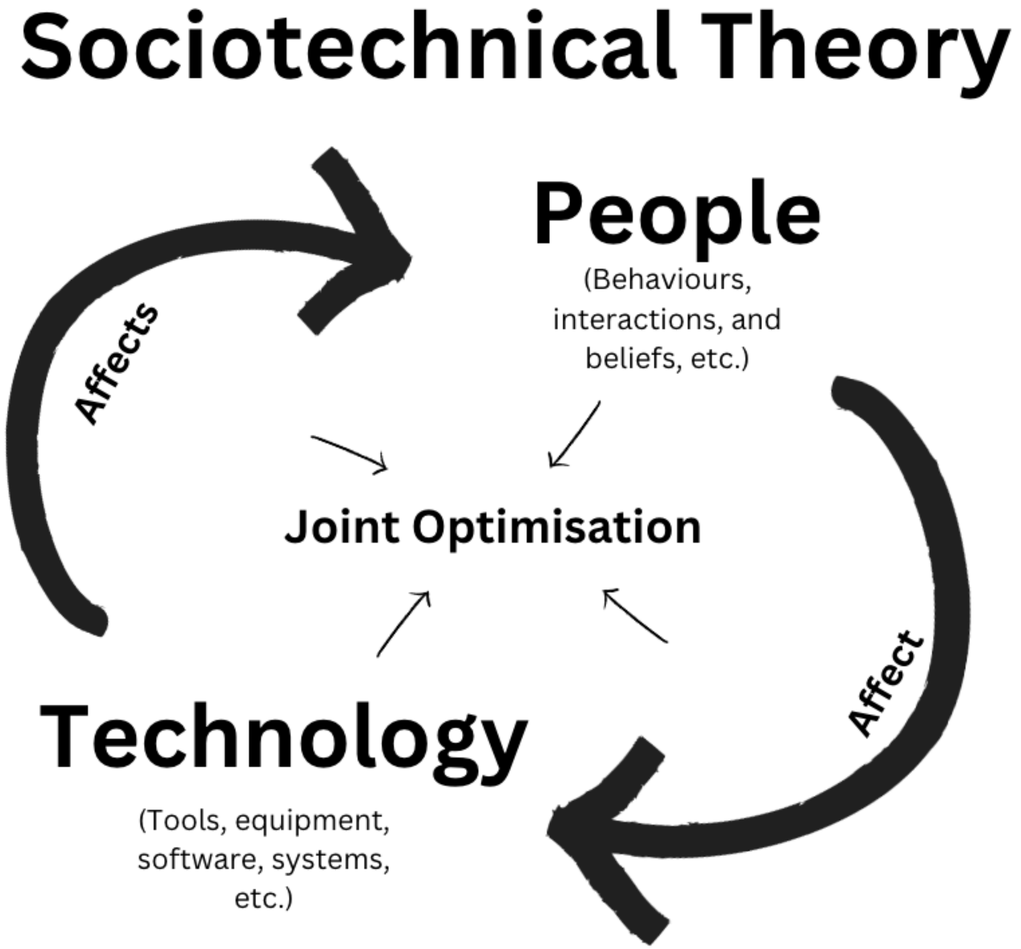
Socio-Technical

Socio-Technical

Cosa significa?

Nel contesto dello sviluppo software, la tecnologia è una parte integrante della vita di uno stakeholder.

Studiare le interazioni di questi stakeholder senza considerare aspetti tecnologici rischia di portare a risultati falsi o incompleti, e dunque, **inutili** [2].



[2] Hoda, Rashina. "Socio-technical grounded theory for software engineering." IEEE Transactions on Software Engineering 48.10 (2021): 3808-3832.

Socio-Technical

Anti-pattern

Anti-pattern

Cosa significa?

Un *anti-pattern* è una soluzione comunemente adottata per un problema ricorrente, che però è tipicamente non efficace, rischiando di risultare invece controproducente.

Social

Anti-pattern

Social anti-pattern fa riferimento a comportamenti e pratiche negativi o controproducenti all'interno di sistemi sociali, in particolare in contesti organizzativi o di team.

Socio-Technical

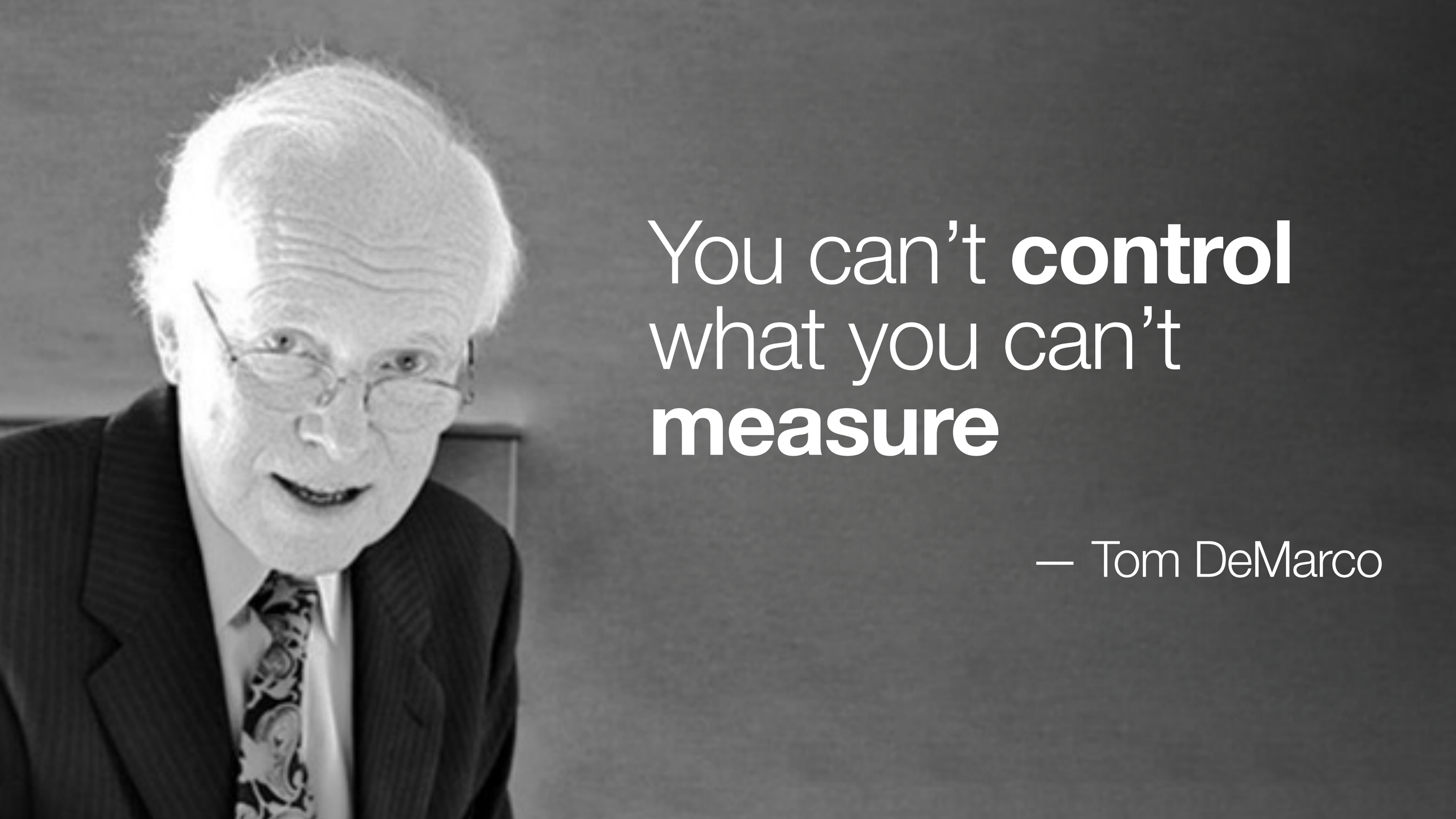
Anti-pattern

Socio-technical anti-pattern fa riferimento a comportamenti e pratiche negativi o controproducenti in sistemi in cui c'è una forte connessione tra persone e tecnologia.

Qual è la differenza?

SOCIAL DEBT

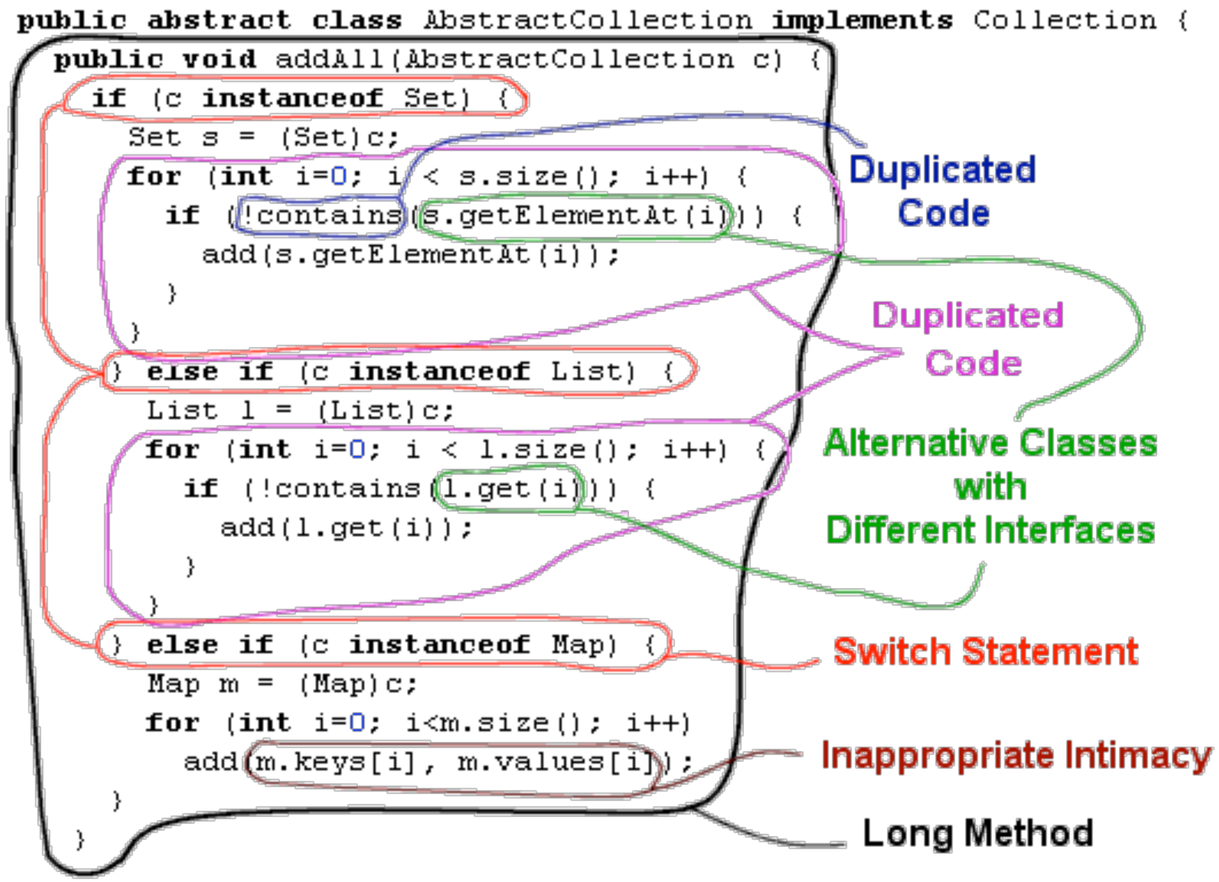
Come possiamo gestirlo?


You can’t **control**
what you can’t
measure
— Tom DeMarco

Code Smell

I Code Smell sono sintomi di scelte progettuali o implementative inadeguate

– Martin Fowler



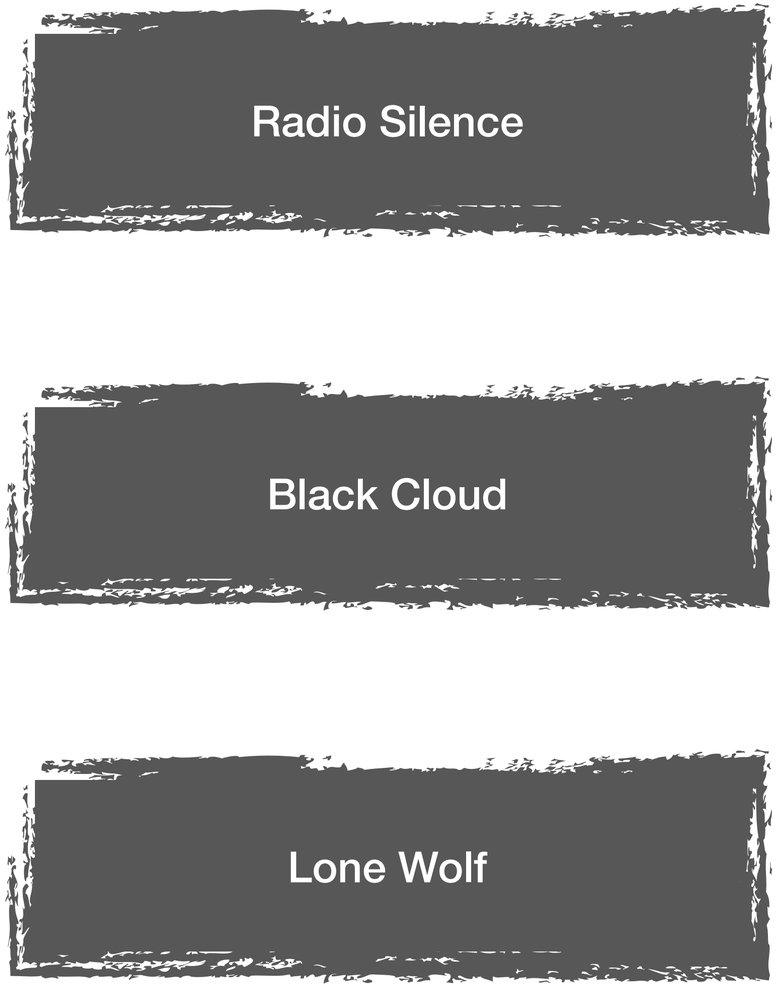
~~Code~~ Community Smell

I Community Smell sono socio-technical anti-pattern presenti nella struttura comunicativa e collaborativa di una comunità software [3].

[3] Caballero-Espinosa, Eduardo, et al. "Community smells—The sources of social debt: A systematic literature review." Information and Software Technology 153 (2023): 107078.

~~Code~~ Community Smell

I Community Smell sono socio-technical anti-pattern presenti nella struttura comunicativa e collaborativa di una comunità software [3].



[3] Caballero-Espinosa, Eduardo, et al. "Community smells—The sources of social debt: A systematic literature review." Information and Software Technology 153 (2023): 107078.

~~Code~~ Co
Smell

I Community Smell s
socio-technical anti-
presenti nella struttu
comunicativa e colla
di una comunità soft

Community Smells - The Sources of Social Debt:
A Systematic Literature Review

Eduardo Caballero-Espinosa[a,b], Jeffrey C. Carver[c], Kimberly Stowers[d]

[a] *Center for Research, Development, and Innovation in Information and Communication Technology (CIDITIC) - Technological University of Panama*
[b] *Computing Systems Engineering Department - Technological University of Panama*
[c] *Department of Computer Science, The University of Alabama, Tuscaloosa, AL, USA*
[d] *Tim Fletcher, Co., San Jose, CA, USA*

Abstract

Context: Social debt describes the accumulation of unforeseen project costs (or potential costs) from sub-optimal software development processes. Community smells are sociotechnical anti-patterns and one source of social debt. Because community smells impact software teams, development processes, outcomes, and organizations, we to understand their impact on software engineering.

Objective: To provide an overview of community smells in social debt, based on published literature, and describe future research.

Method: We conducted a systematic literature review (SLR) to identify properties, understand origins and evolution, and describe the emergence of community smells. This SLR explains the impact of community smells on teamwork and team performance.

Results: We include 25 studies. Social debt describes the impacts of poor socio-technical decisions on work environments, people, software products, and society. For each of the 30 community smells identified as sources of social debt, we provide a detailed description, management approaches, organizational strategies, and mitigation effectiveness. We identify five groups of management approaches: organizational strategies, frameworks, models, tools, and guidelines. We describe 11 common properties of community smells. We develop the *Community Smell Stages Framework* to concisely

Radio Silence

Black Cloud

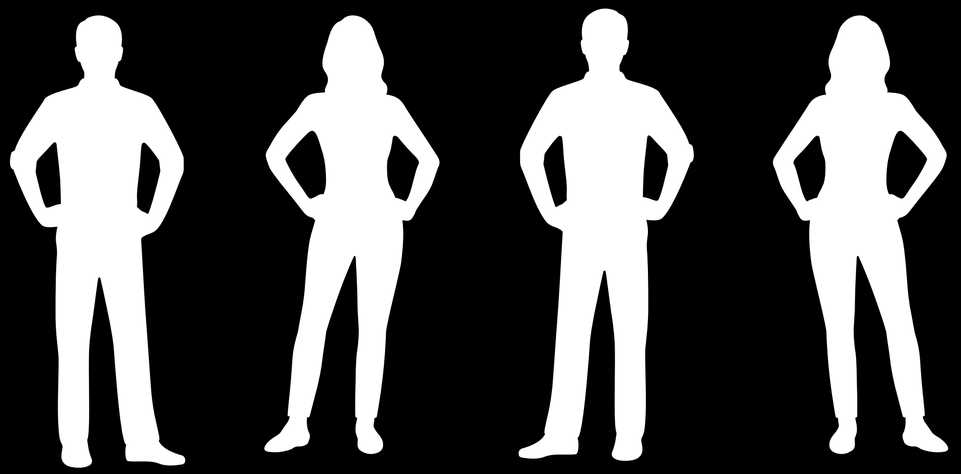
Lone Wolf

[3] Caballero-Espinosa, Eduardo, et al. "Community smells—The sources of social debt: A systematic literature review." Information and Software Technology 153 (2023): 107078.

SOCIAL DEBT

Sono dei costi aggiuntivi di un progetto software che derivano da ***socio-technical anti-pattern*** non previsti e non gestiti all'interno della comunità software [1].

I Community Smell—ossia, **socio-technical anti-pattern** nella comunità di sviluppo—sono precursori di social debt.



[1] Tamburri, Damian A., et al. "What is social debt in software engineering?." 2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE). IEEE, 2013.

IMMAGINA!

Scenario di esempio

Avete passato la vostra intera vita a studiare, lavorare sodo e a scalare la gerarchia aziendale.

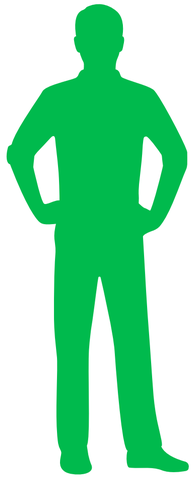
Dopo anni di sacrifici, finalmente il vostro capo vi ha promossi.

Ora siete dei MANAGER!

Scenario di Esempio Team di sviluppo

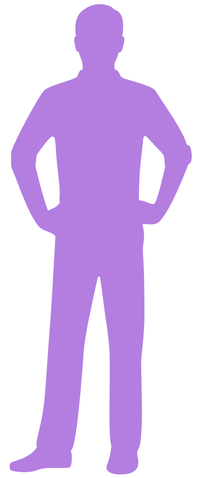
Tonio Crudo

The Manager

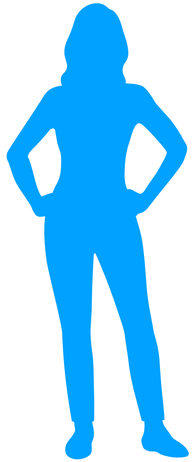


Gilberto

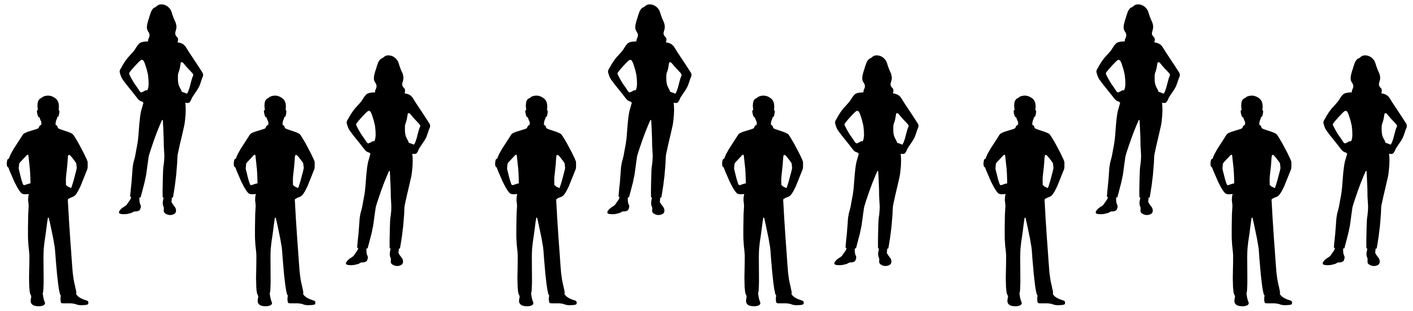
Giulia



Giammaria



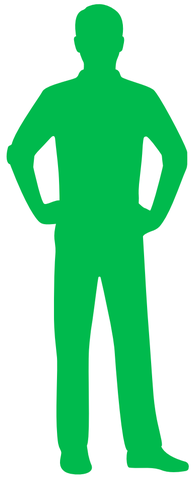
Viviana



Scenario di Esempio Team di sviluppo

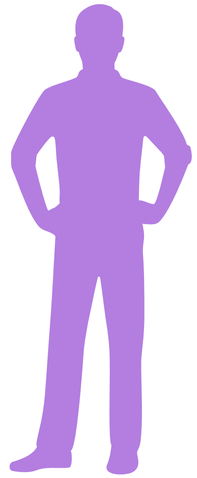
Tonio Crudo

The Manager

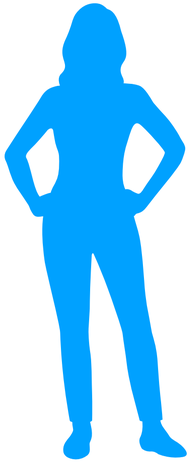


Gilberto

Giulia



Giammaria

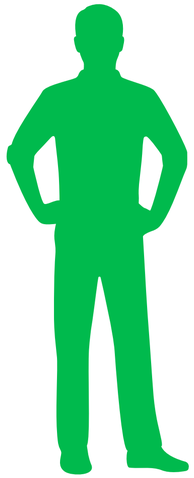


Viviana

ESSE 4

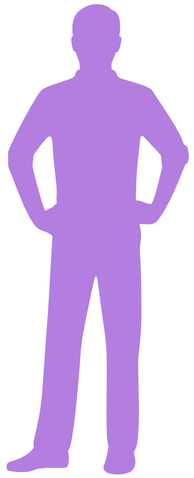
Scenario di Esempio

Tonio Crudo

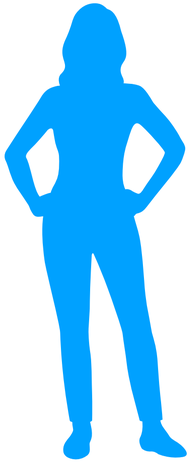


Gilberto

Giulia



Giammaria

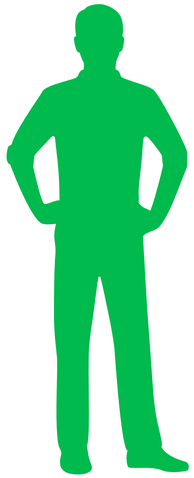


Viviana

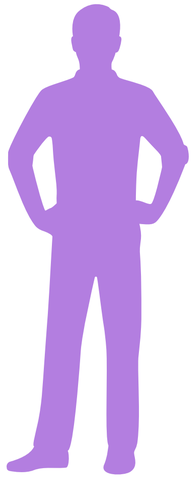
Scenario di Esempio
Comunicazione nel Team

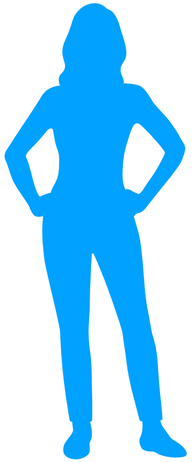
ESSE 4


Tonio Crudo


Gilberto


Giulia


Giammaria

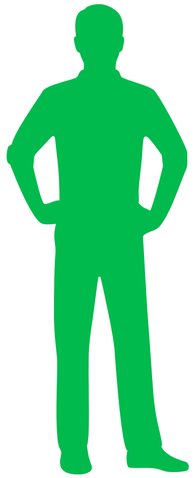

Viviana

slack

Microsoft Teams

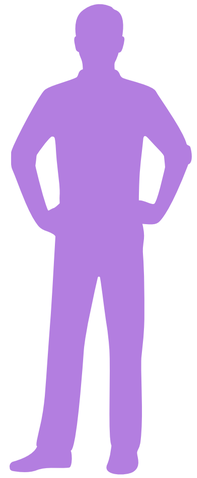
Scenario di Esempio
Comunicazione nel Team

Tonio Crudo

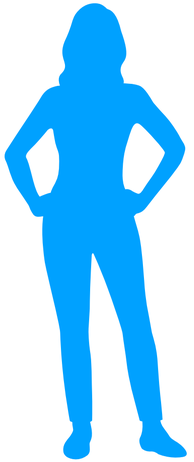


Gilberto

Giulia



Giammaria

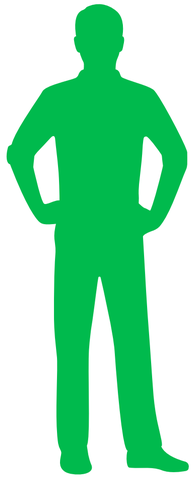


Viviana

Dove si trova la key per accedere al DB?

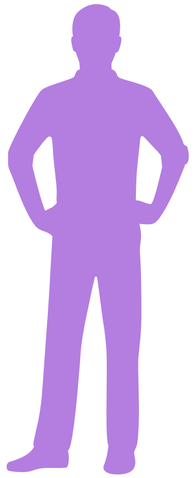
Scenario di Esempio
Comunicazione nel Team

Tonio Crudo

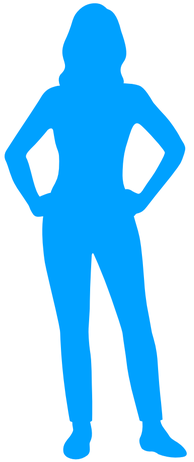


Gilberto

Giulia



Giammaria



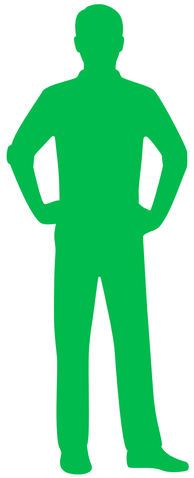
Viviana

Dove si trova la key per accedere al DB?

Chi è che si sta occupando del design pattern?

Scenario di Esempio
Comunicazione nel Team

Tonio Crudo

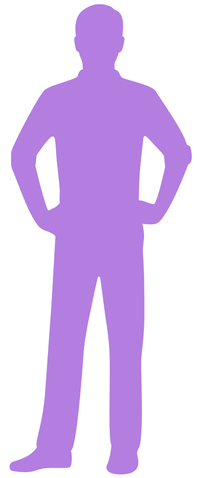


Gilberto

Dove si trova la key per accedere al DB?

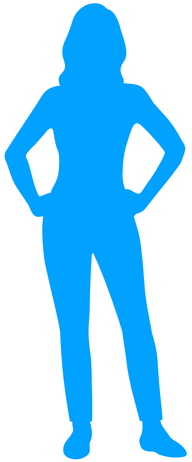
Giulia

Chi è che si sta occupando del design pattern?



Giammaria

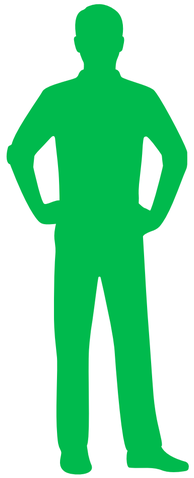
A che ora è il meeting domani?



Viviana

Scenario di Esempio
Comunicazione nel Team

Tonio Crudo

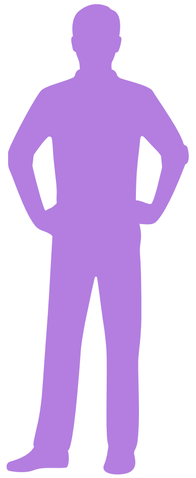


Gilberto

Dove si trova la key per accedere al DB?

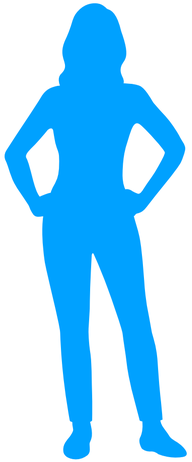
Giulia

Chi è che si sta occupando del design pattern?



Giammaria

A che ora è il meeting domani?

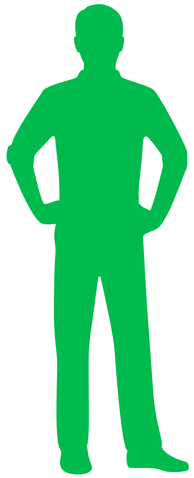


Viviana

Ma quindi il nostro è un MVC?

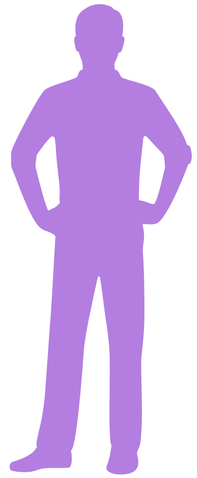
Scenario di Esempio
Comunicazione nel Team

Tonio Crudo

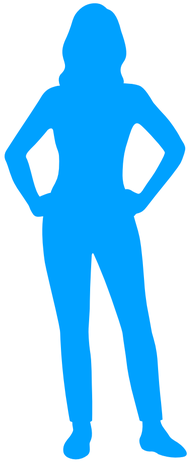


Gilberto

Giulia



Giammaria

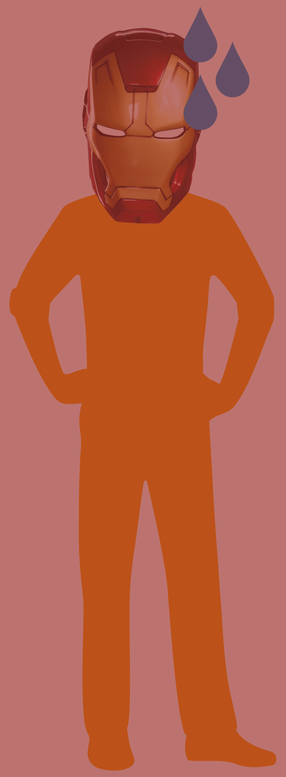


Viviana

Presto, vi rendete conto che la troppa formalità nella comunicazione, con una conseguente diminuzione delle occasioni per scambiare informazioni in maniera informale, ha portato ad una perdita di conoscenza globale, risultando in un overhead di comunicazione verso il manager.

Scenario di Esempio
Comunicazione nel Team

ESSE 4

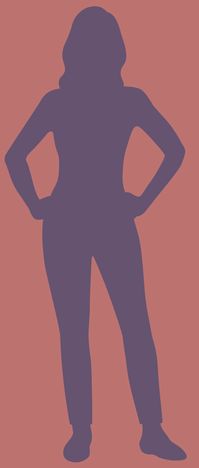


Tonio Crudo

Alberto

Iulia

Gianmaria



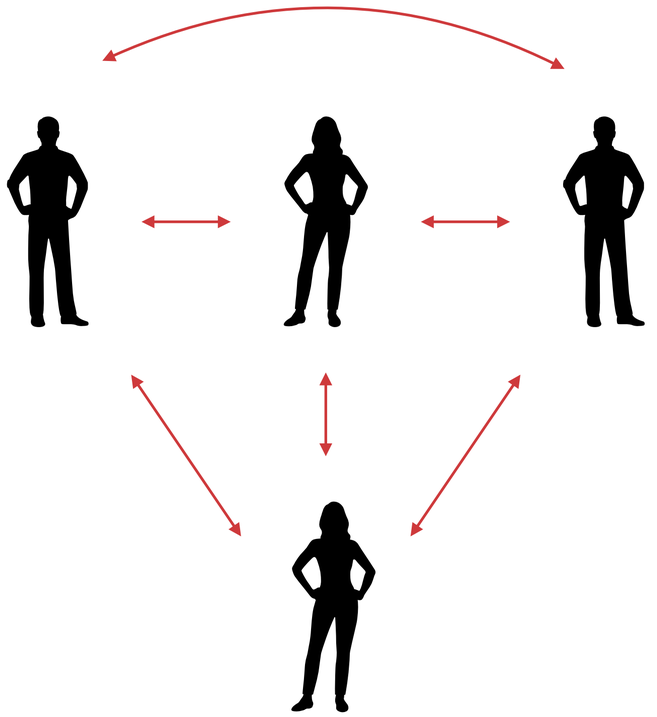
Viviana

BLACK CLOUD

Presto, vi rendete conto che la troppa formalità nella comunicazione, con una conseguente diminuzione delle occasioni per scambiare informazioni in maniera informale, ha portato ad una perdita di conoscenza globale, risultando in un overhead di comunicazione verso il manager.

Black Cloud

Un esempio di Community Smell



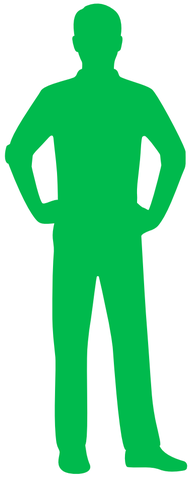
Questo smell si presenta quando le organizzazioni non mettono in condizione i compagni di team di interagire socialmente e comunicare tra loro in maniera efficace.

Per cui, le condizioni in cui lavorano non supportano lo scambio di conoscenza (ad esempio, esperienza professionale o comprensione dei progetti in corso) durante lo sviluppo software [3].

[3] Caballero-Espinosa, Eduardo, et al. "Community smells—The sources of social debt: A systematic literature review." Information and Software Technology 153 (2023): 107078.

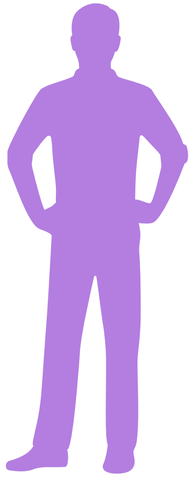
Scenario di Esempio Lavoro di gruppo

Tonio Crudo



Gilberto

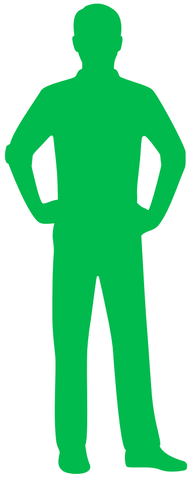
Giulia



Giammaria

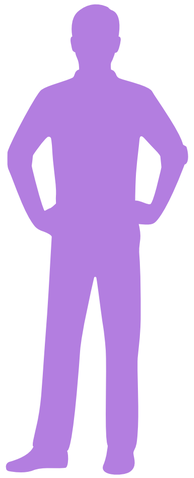
Scenario di Esempio Lavoro di gruppo

Tonio Crudo



Gilberto

Giulia



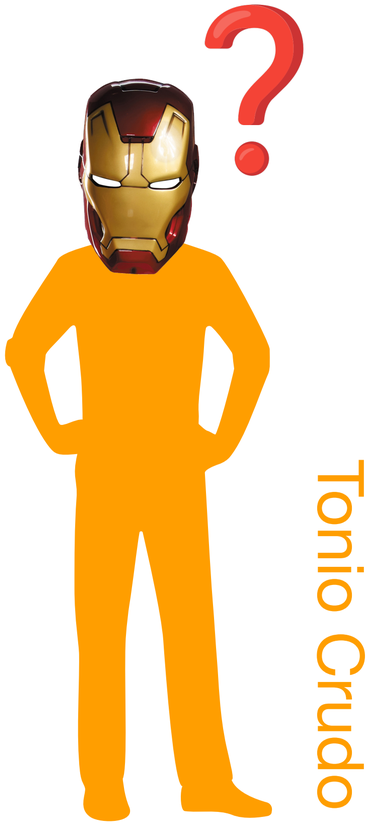
Giammaria

```
IDENTIFICATION DIVISION.
       PROGRAM-ID. ADD_NUMBERS.
       DATA DIVISION.
       FILE SECTION.
       WORKING-STORAGE SECTION.
       01  FIRST-NUMBER    PICTURE IS 99.
       01  SECOND-NUMBER   PICTURE IS 99.
       01  RESULT          PICTURE IS 9999.
       PROCEDURE DIVISION.

       MAIN-PROCEDURE.
           DISPLAY "Here is the first Number "
           MOVE 8 TO FIRST-NUMBER
           DISPLAY FIRST-NUMBER

           DISPLAY "Let's add 20 to that number."
           ADD 20 TO FIRST-NUMBER
           DISPLAY FIRST-NUMBER

           DISPLAY "Create a second variable"
           MOVE 30 TO SECOND-NUMBER
           DISPLAY SECOND-NUMBER

           *>COMMENT: COMPUTE THE TWO NUMBER AND PLACE INTO RESULT*
           COMPUTE RESULT = FIRST-NUMBER +  SECOND-NUMBER.

           DISPLAY "The result is:".
           DISPLAY RESULT.
         STOP RUN.
END PROGRAM ADD_NUMBERS.

```

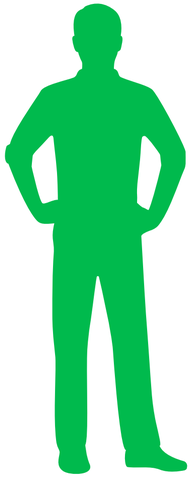
Scenario di Esempio Lavoro di gruppo


```
IDENTIFICATION DIVISION.
       PROGRAM-ID. ADD_NUMBERS.
       DATA DIVISION.
       FILE SECTION.
       WORKING-STORAGE SECTION.
       01  FIRST-NUMBER    PICTURE IS 99.
       01  SECOND-NUMBER   PICTURE IS 99.
       01  RESULT          PICTURE IS 9999.
       PROCEDURE DIVISION.

       MAIN-PROCEDURE.
           DISPLAY "Here is the first Number "
           MOVE 8 TO FIRST-NUMBER
           DISPLAY FIRST-NUMBER

           DISPLAY "Let's add 20 to that number."
           ADD 20 TO FIRST-NUMBER
           DISPLAY FIRST-NUMBER

           DISPLAY "Create a second variable"
           MOVE 30 TO SECOND-NUMBER
           DISPLAY SECOND-NUMBER

           *>COMMENT: COMPUTE THE TWO NUMBER AND PLACE INTO RESULT*
           COMPUTE RESULT = FIRST-NUMBER +  SECOND-NUMBER.

           DISPLAY "The result is:".
           DISPLAY RESULT.
         STOP RUN.
END PROGRAM ADD_NUMBERS.

```

Scenario di Esempio Lavoro di gruppo

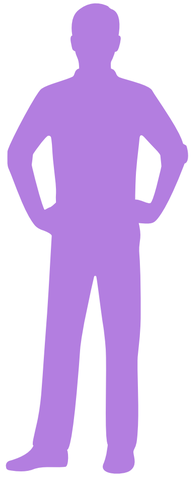


Tonio Crudo

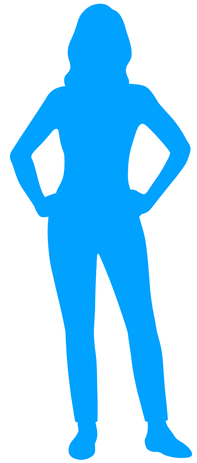


Gilberto

Giulia



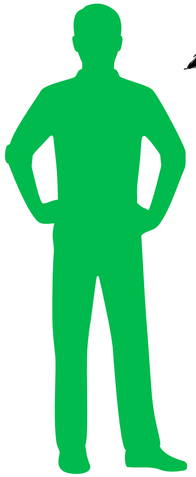
Giammaria



Viviana

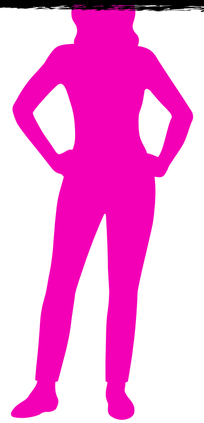
Scenario di Esempio Lavoro di gruppo

Tonio Crudo



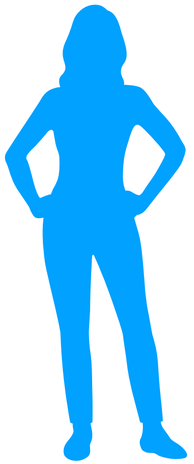
Gilberto

Ha riscritto tutto il mio codice senza chiedermelo!

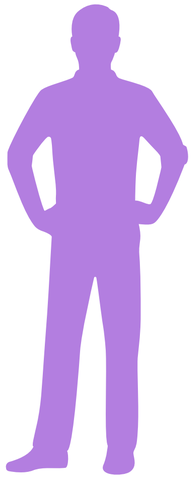


Gisella

Continua a insistere che il sito dovrebbe essere azzurro quando abbiamo scelto per il rosso!



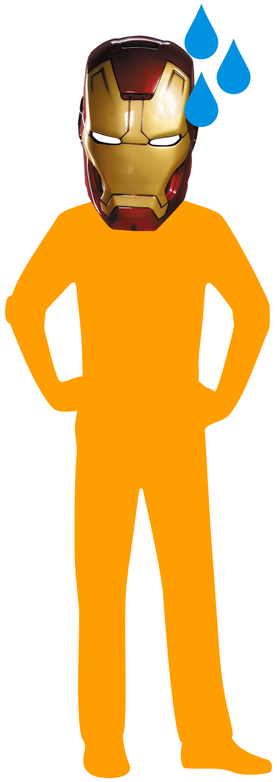
Viviana



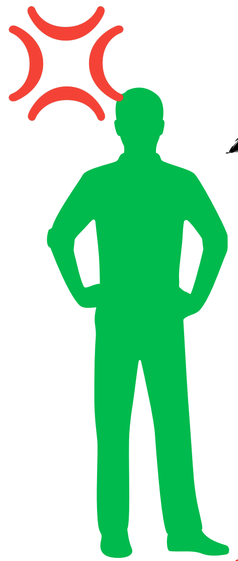
Giammaria

Perché continua a usare il PascalCase se abbiamo deciso di fare tutto in camelCase?!?

Scenario di Esempio Lavoro di gruppo



Tonio Crudo

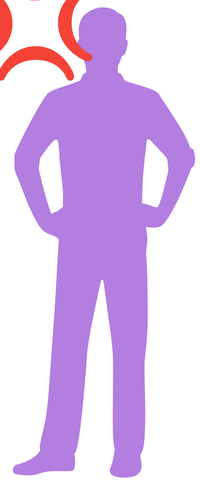


Gilberto

Ha riscritto tutto il mio codice senza chiedermelo!

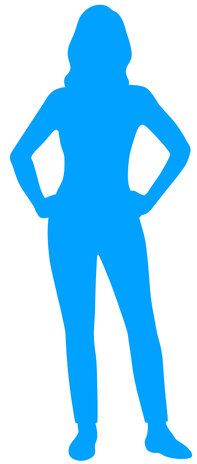
Gisella

Continua a insistere che il sito dovrebbe essere azzurro quando abbiamo scelto per il rosso!



Giammaria

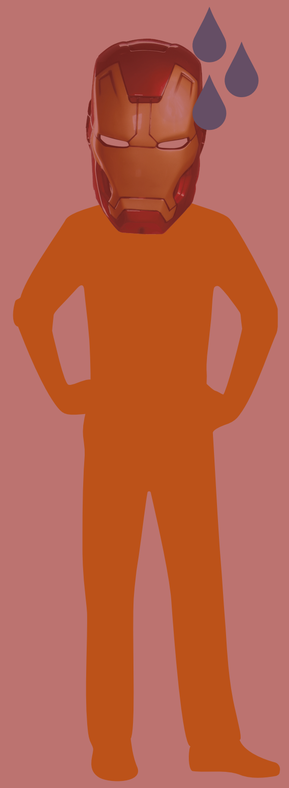
Perché continua a usare il PascalCase se abbiamo deciso di fare tutto in camelCase?!?



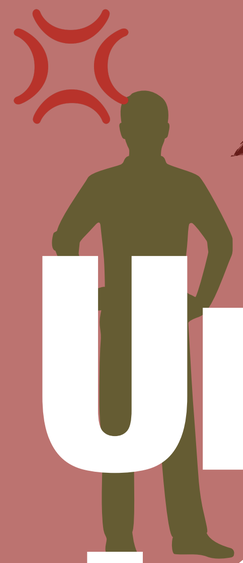
Viviana

I don’t care!

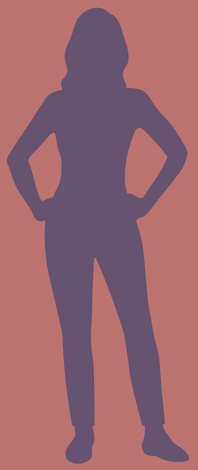
Scenario di Esempio Lavoro di gruppo



Tonio Crudo



Riccardo Riammaria



Viviana

Ha riscritto tutto il mio codice senza chiedermelo!

Perché continua a usare il PascalCase se abbiamo deciso di fare tutto in camelCase?!?

I don't care!

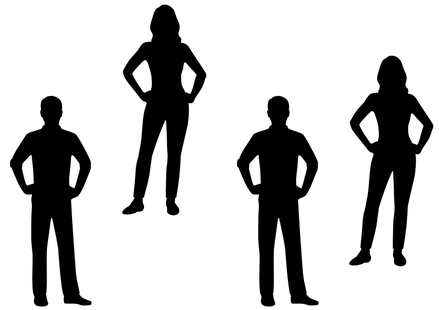
Unmanaged LONE WOLF

Lone Wolf

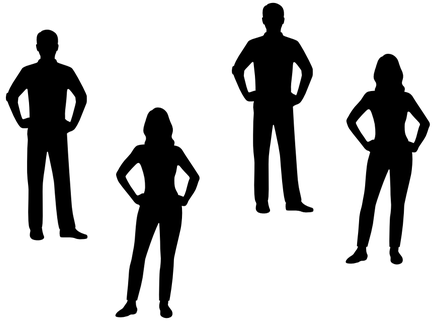
Un esempio di Community Smell

È uno scenario in cui un membro del team si isola e lavora da solo, evitando la comunicazione con i suoi colleghi.

Una situazione simile porta a task eseguiti in isolamento e perdita di controllo sul tracciamento del progetto, oltre che a un generale malcontento del gruppo [3].



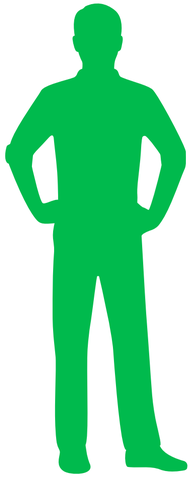
Lone Wolf



[3] Caballero-Espinosa, Eduardo, et al. "Community smells—The sources of social debt: A systematic literature review." Information and Software Technology 153 (2023): 107078.

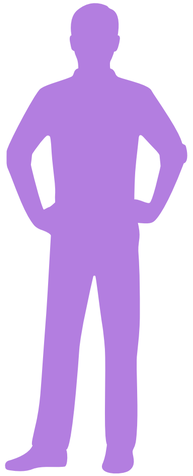
Scenario di Esempio

Tonio Crudo

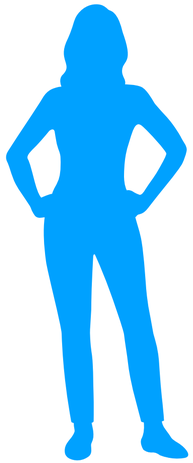


Gilberto

Giulia



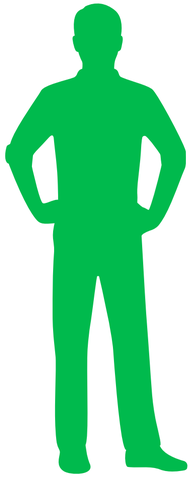
Giammaria



Viviana

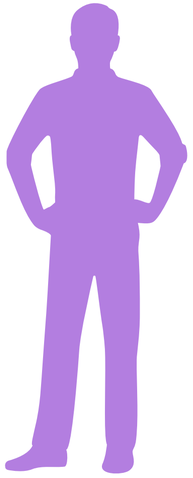
Scenario di Esempio

Tonio Crudo

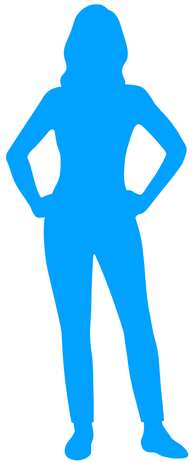


Gilberto

Giulia

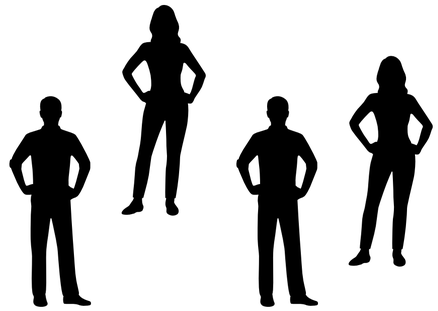


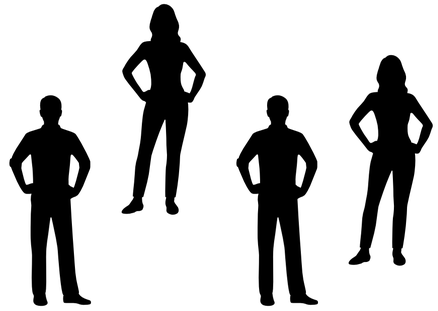
Giammaria



Viviana

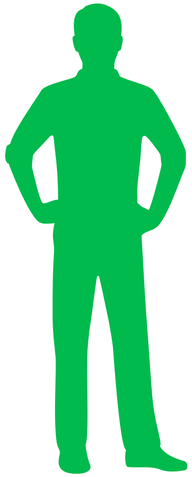
Ruggero





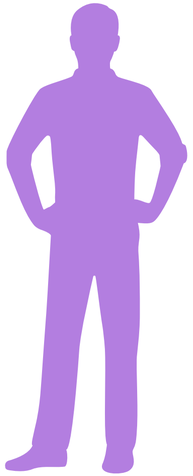
Scenario di Esempio

Tonio Crudo

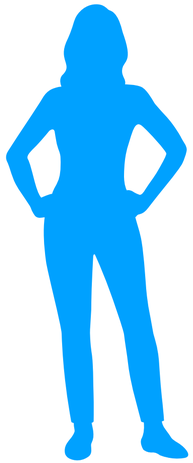


Gilberto

Giulia

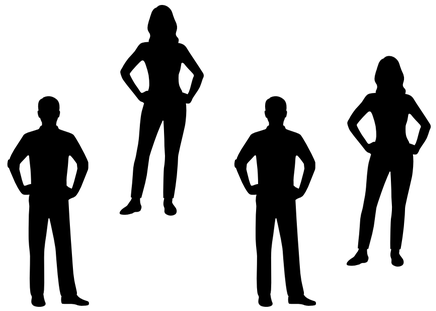


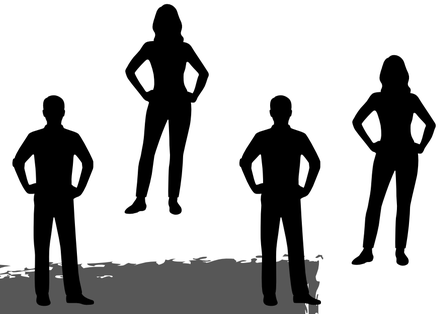
Giammaria



Viviana

Ruggero



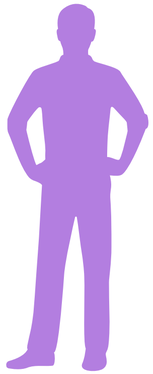


É necessario definire un modo per far comunicare i due team di sviluppo!

Scenario di Esempio

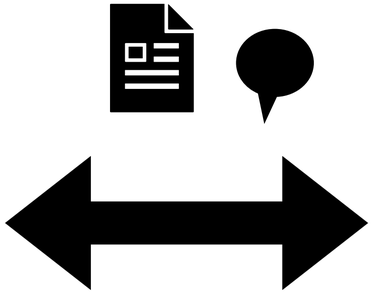
Tonio Crudo

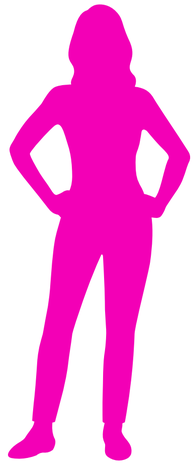
Gilberto



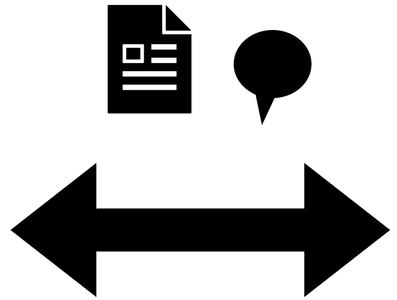
Giammaria

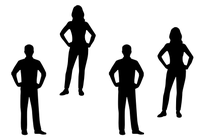
Viviana

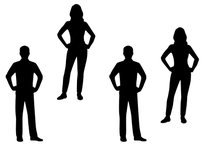




Giulia



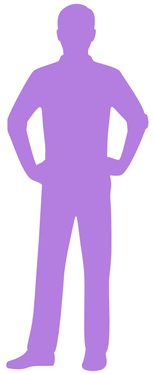




Scenario di Esempio

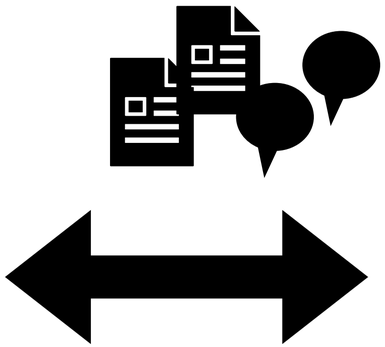
Tonio Crudo

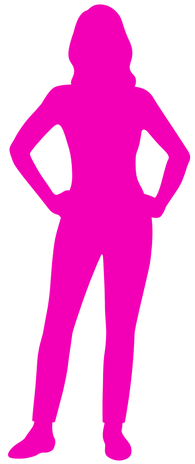
Gilberto



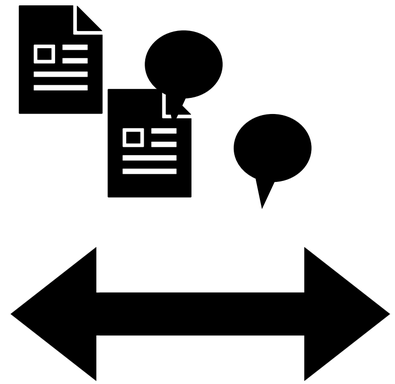
Giammaria

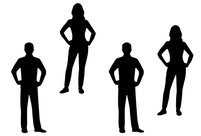
Viviana

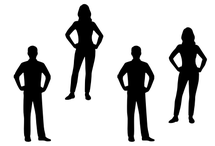




Giulia



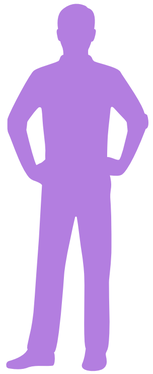




Scenario di Esempio

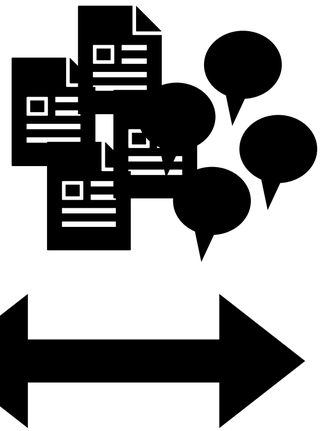
Tonio Crudo

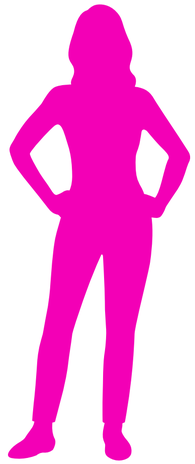
Gilberto



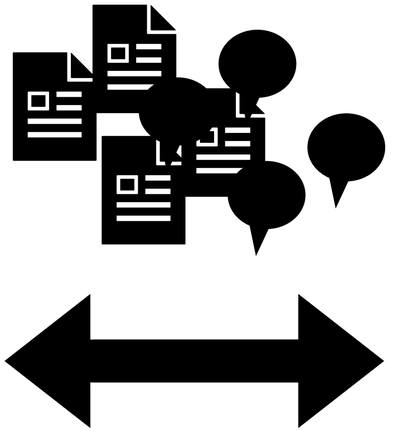
Giammaria

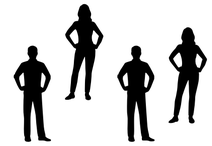
Viviana





Giulia

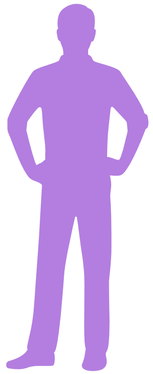




Scenario di Esempio

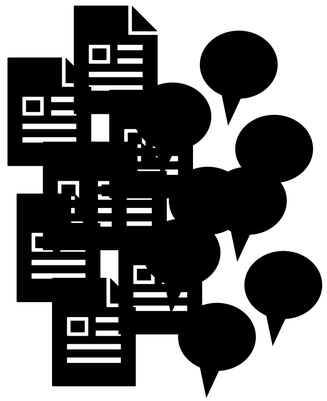
Tonio Crudo

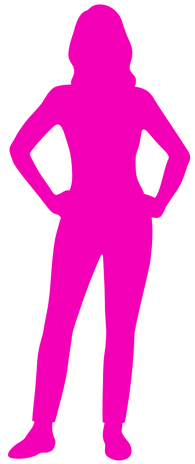
Gilberto



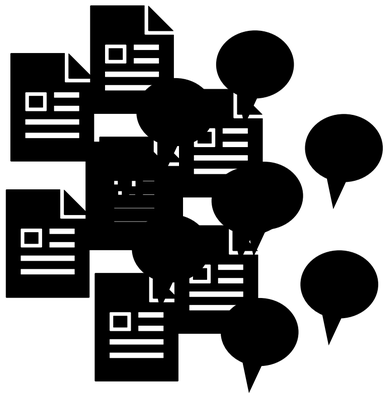
Giammaria

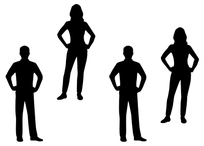
Viviana





Giulia

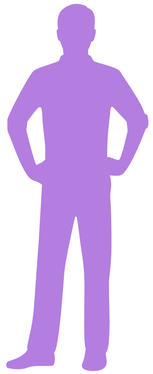




Scenario di Esempio

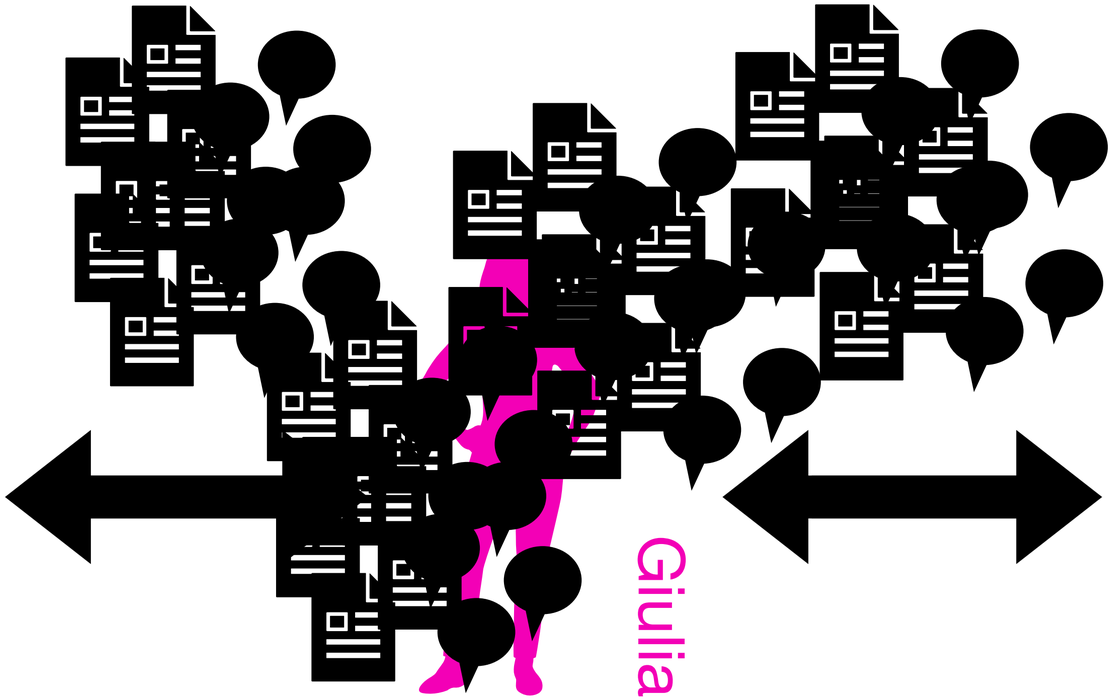
Tonio Crudo

Gilberto

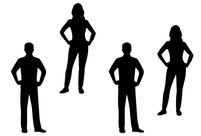


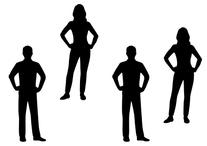
Giammaria

Viviana



Giulia

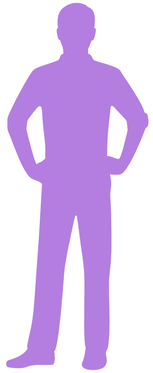




Scenario di Esempio

Tonio Crudo

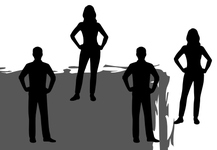
Gilberto



Giammaria

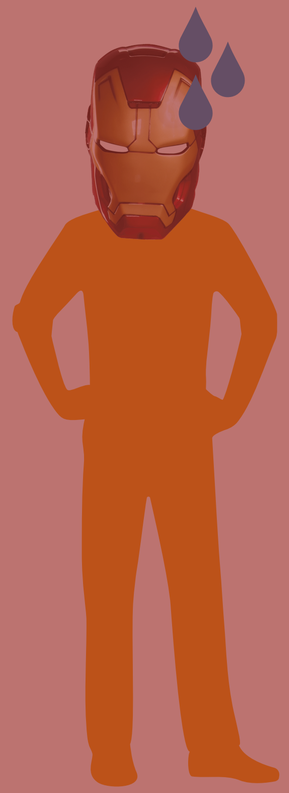
Viviana

Giulia



La povera Giulia non è in grado di gestire tutto questo scambio di informazioni, inevitabilmente risultando in una perdita di alcune di esse.

Scenario di Esempio



Tonio Crudo

Gilberto Giammaria

Viviana

Giulia

RADIO SILENCE

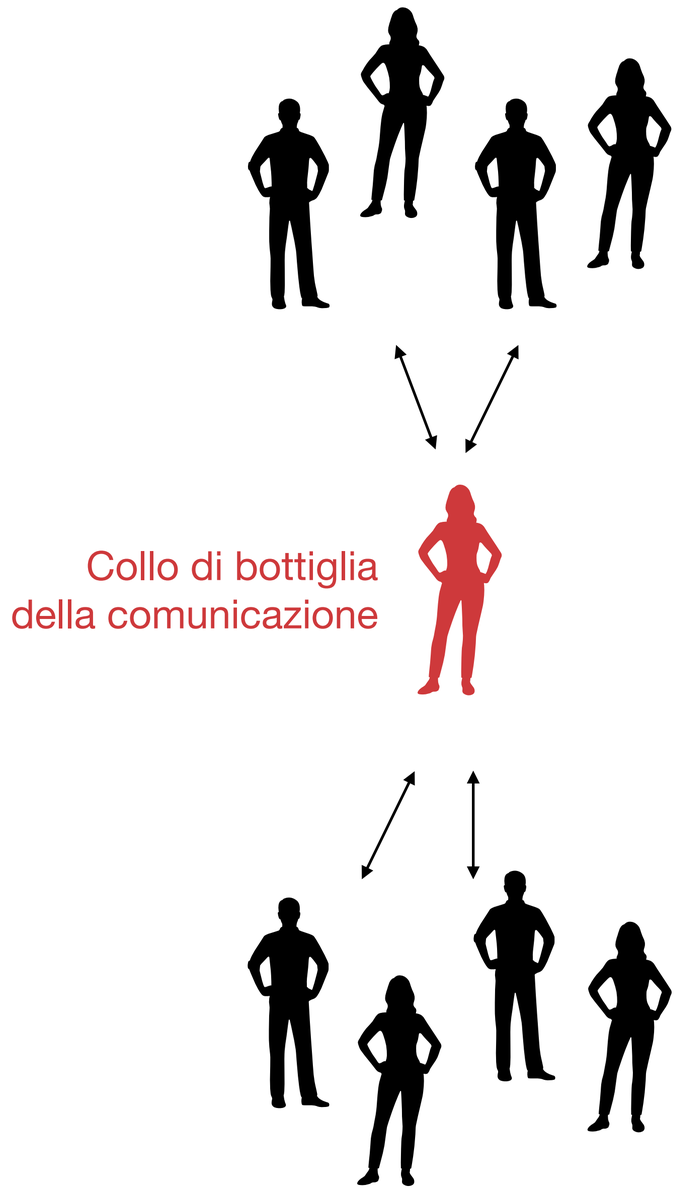
La povera Giulia non è in grado di gestire tutto questo scambio di informazioni, inevitabilmente risultando in una perdita di alcune di esse.

Radio Silence

Un esempio di Community Smell

È uno scenario in cui i capi e i membri del team eseguono i task seguendo un’organizzazione estremamente formale e complessa. In questo modo, le strutture comunicative non permettono uno scambio efficiente di informazioni tra i team.

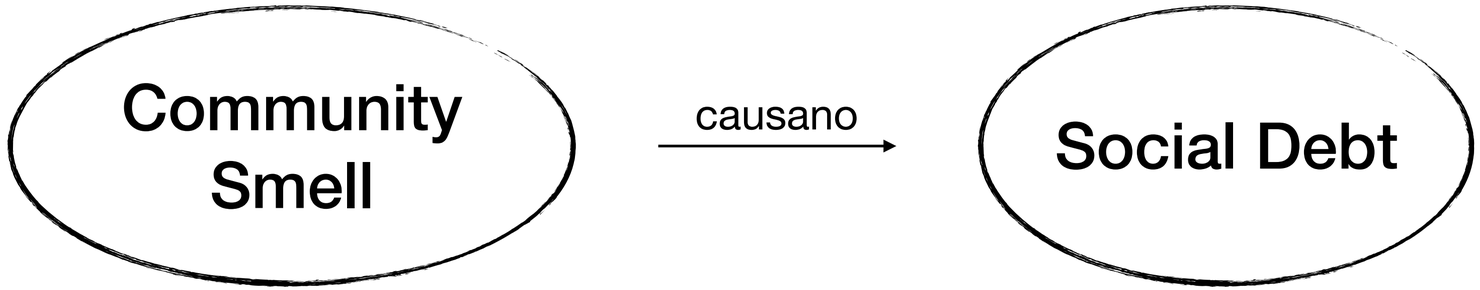
Ad esempio, si può verificare quando è una sola persona a fungere da intermediario per le comunicazioni tra diversi team, provocando un sovraccarico comunicativo e ingenti ritardi [3].



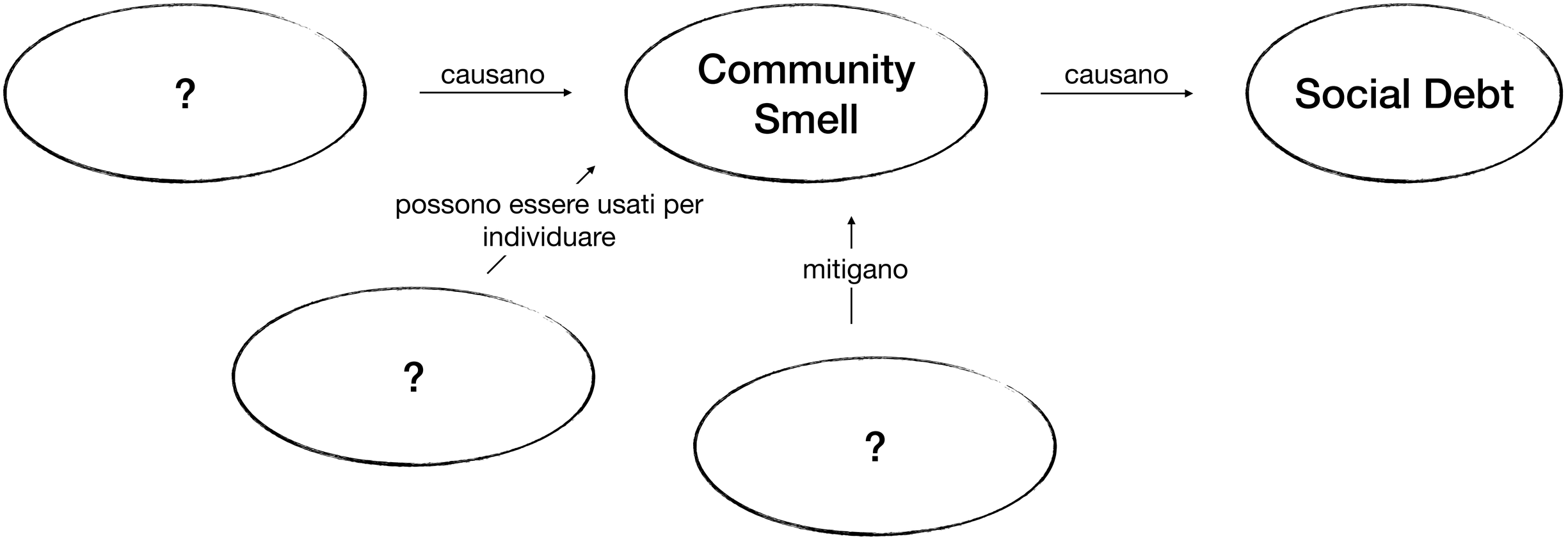
[3] Caballero-Espinosa, Eduardo, et al. "Community smells—The sources of social debt: A systematic literature review." Information and Software Technology 153 (2023): 107078.


VICTORY

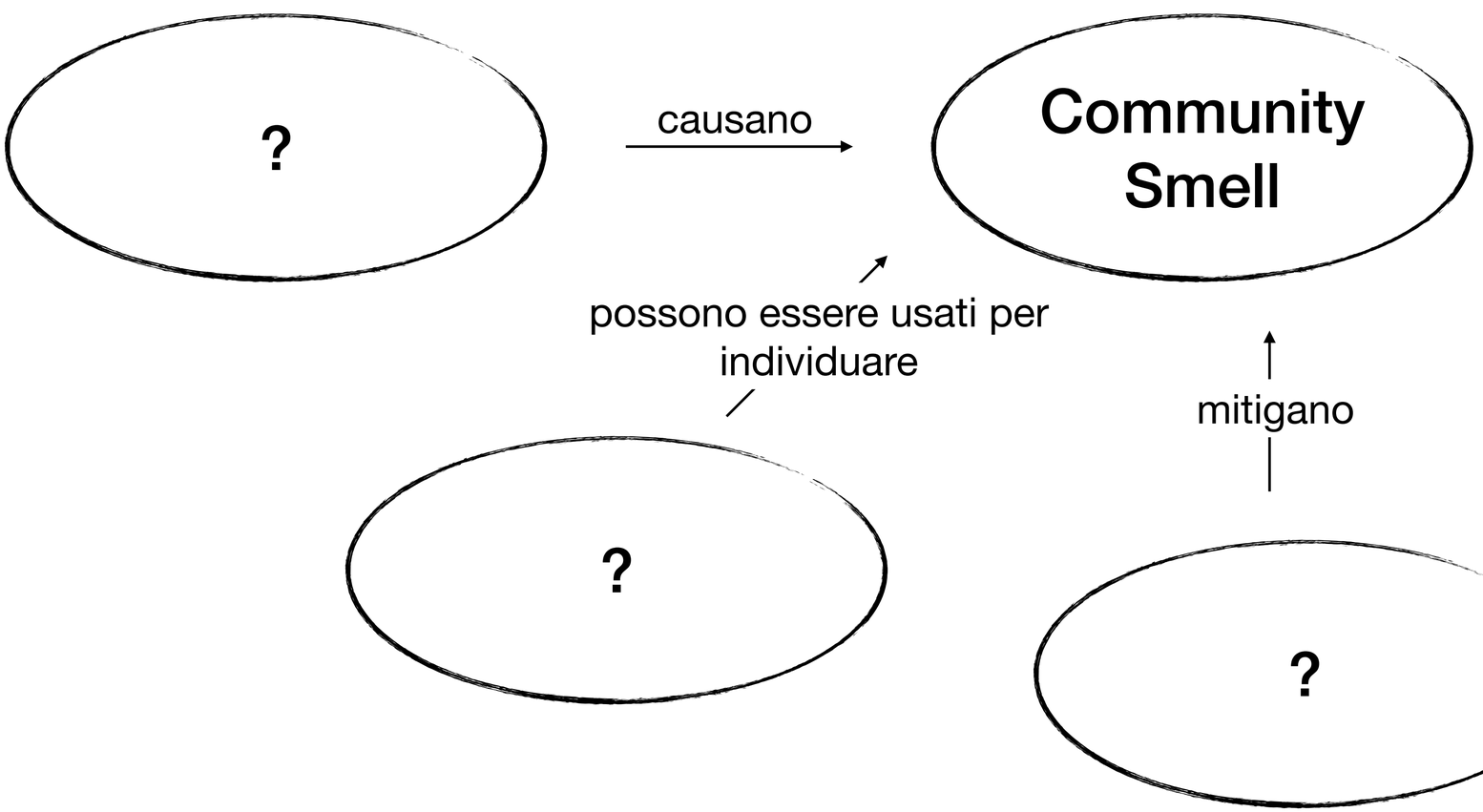
Community Smell & Social Debt



Community Smell & Social Debt

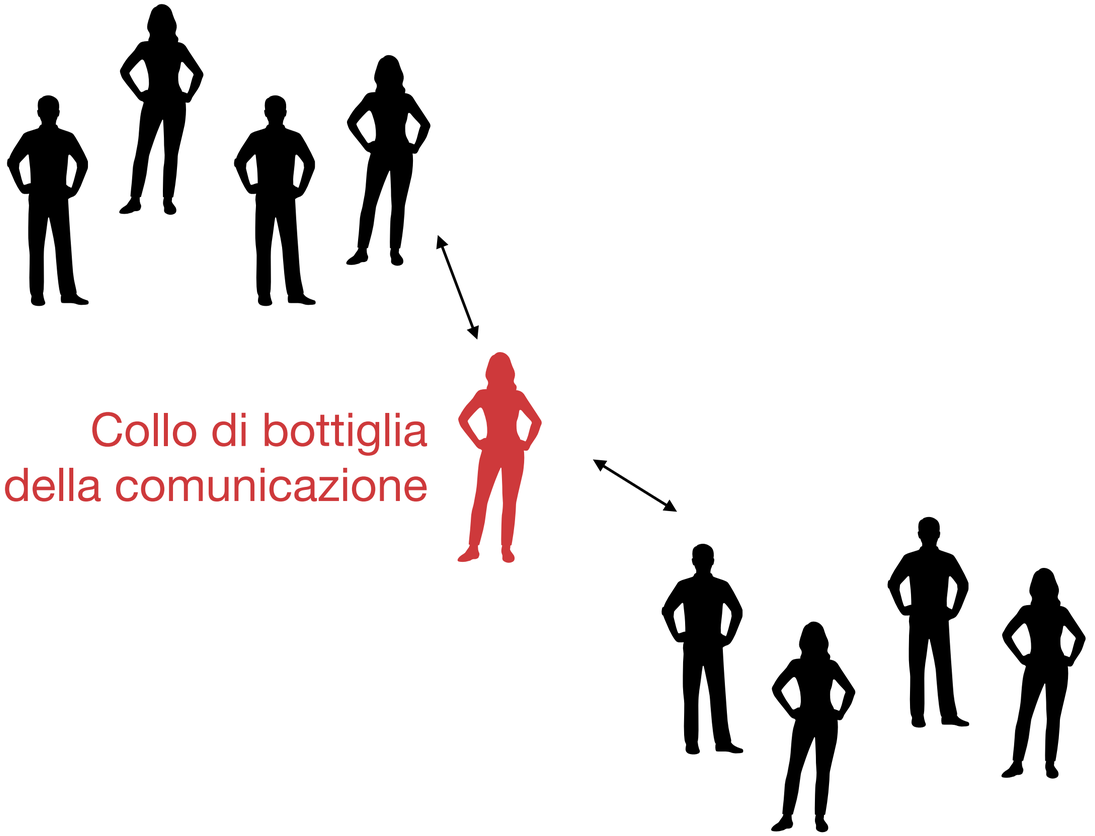


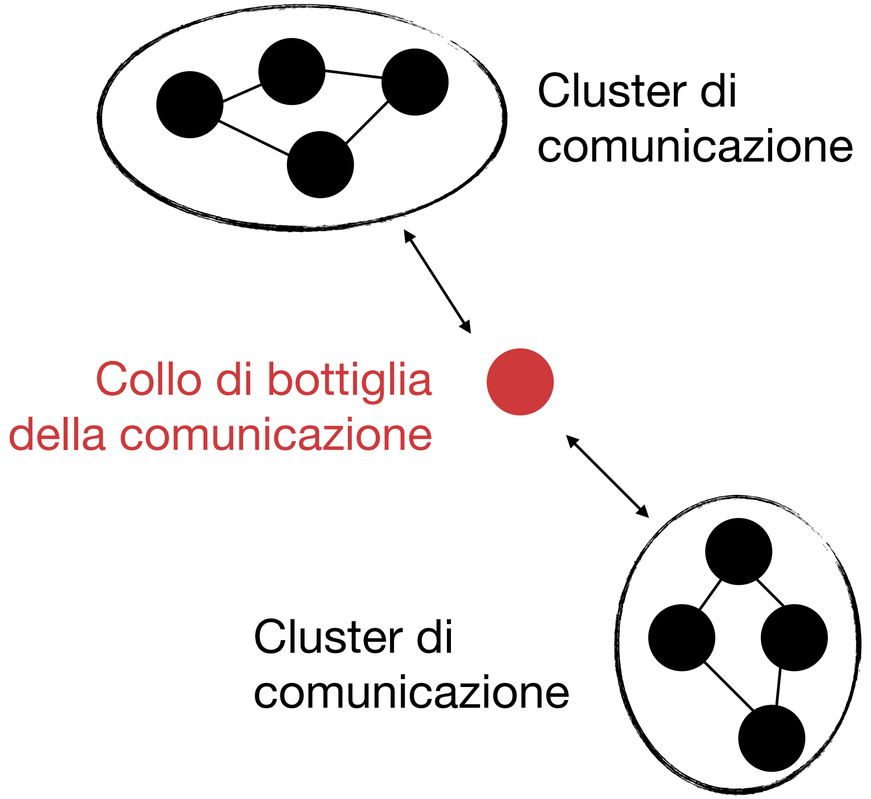
La Ricerca sui Community Smell



Radio Silence

Rappresentazione degli smell come grafi





[3] Caballero-Espinosa, Eduardo, et al. "Community smells—The sources of social debt: A systematic literature review." Information and Software Technology 153 (2023): 107078.

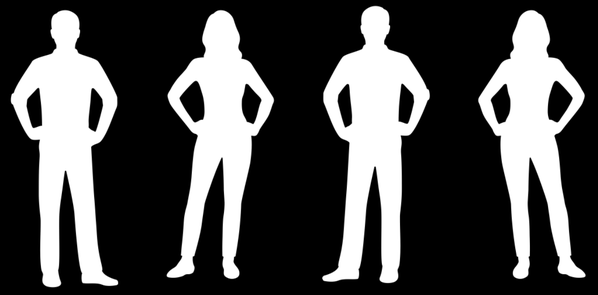
Conclusioni

Che cosa avete (presumibilmente) imparato?

SOCIAL DEBT

Sono dei costi aggiuntivi di un progetto software che derivano da ***socio-technical anti-pattern*** non previsti e non gestiti all'interno della comunità software [1].

I Community Smell—ossia, **socio-technical anti-pattern** nella comunità di sviluppo—sono precursori di social debt.



[1] Tamburri, Damian A., et al. "What is social debt in software engineering?." 2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE). IEEE, 2013.

- Cos'è il Social Debt
- Cos'è un socio-technical anti-pattern
- Cosa sono i community smell
- Qual è la correlazione tra community smell esocial debt

Materiale esterno di ricerca

[1] Tamburri, Damian A., et al. "What is social debt in software engineering?." 2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE). IEEE, 2013.

[2] Hoda, Rashina. "Socio-technical grounded theory for software engineering." IEEE Transactions on Software Engineering 48.10 (2021): 3808-3832.

[3] Caballero-Espinosa, Eduardo, et al. "Community smells—The sources of social debt: A systematic literature review." Information and Software Technology 153 (2023): 107078.

[4] Palomba, Fabio, et al. "Beyond technical aspects: How do community smells influence the intensity of code smells?." IEEE transactions on software engineering 47.1 (2018): 108-129.

[5] Catolino, Gemma, et al. "Gender diversity and women in software teams: How do they affect community smells?." 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS). IEEE, 2019.

[6] Lambiase, Stefano, et al. "Good fences make good neighbours? on the impact of cultural and geographical dispersion on community smells." Proceedings of the 2022 ACM/IEEE 44th International Conference on Software Engineering: Software Engineering in Society. 2022.

[7] Almarimi, Nuri, Ali Ouni, and Mohamed Wiem Mkaouer. "Learning to detect community smells in open source software projects." Knowledge-Based Systems 204 (2020): 106201.