TeamLead Conf · April 29–30, 2021

# The SOLID Team Lead

Management fundamentals for engineers

TeamLead Conf 2021



Audience For whom

# The engineer stepping into management

Who this talk is for

- You are a strong engineer
- You are new to management
- You already lead a team, or are about to
- You want to succeed in the new role

Part 1

01

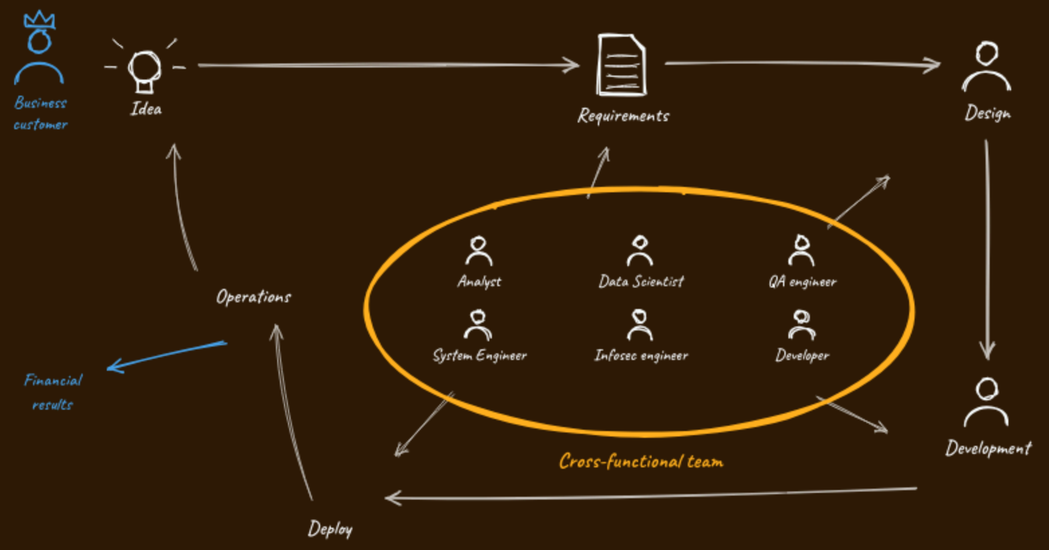
# The team

Who carries an idea through to a result

# The team closes the loop from idea to financial result

A modern cross-functional development team

- Roles: Analyst, Data Scientist, System Engineer
- ...QA engineer, Developer, Infosec engineer
- Loop: idea → requirements → design → development
- ...deploy → operations → financial results



# The team lead’s job is to make the team an excellent one

The customer’s ideal team

- The business customer brings an idea
- In return they expect financial results
- Between them sits one «development team»
- Making that team excellent is the team lead’s job

OOP 4 principles

# Let us start solving it with four familiar words

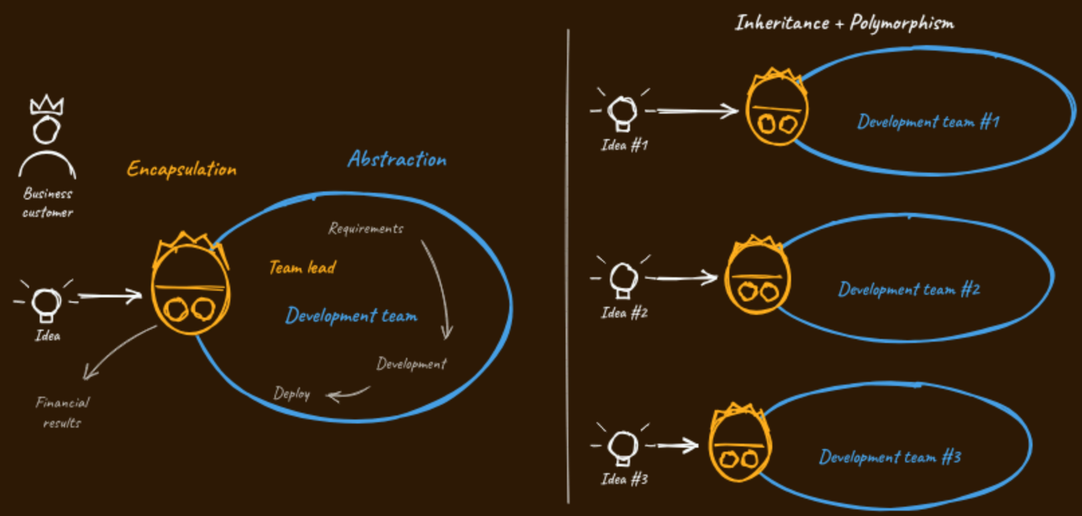
OOP principles

- Abstraction — only what matters from the outside
- Encapsulation — details stay hidden inside
- Inheritance — a shared shape gets reused
- Polymorphism — one interface, many implementations

# The team is an object, and the team lead is its interface

The same four words applied to a team

- Abstraction: the customer sees a team, not six roles
- Encapsulation: the team lead stands between them
- Inheritance: the same team shape for every idea
- Polymorphism: three teams, three implementations

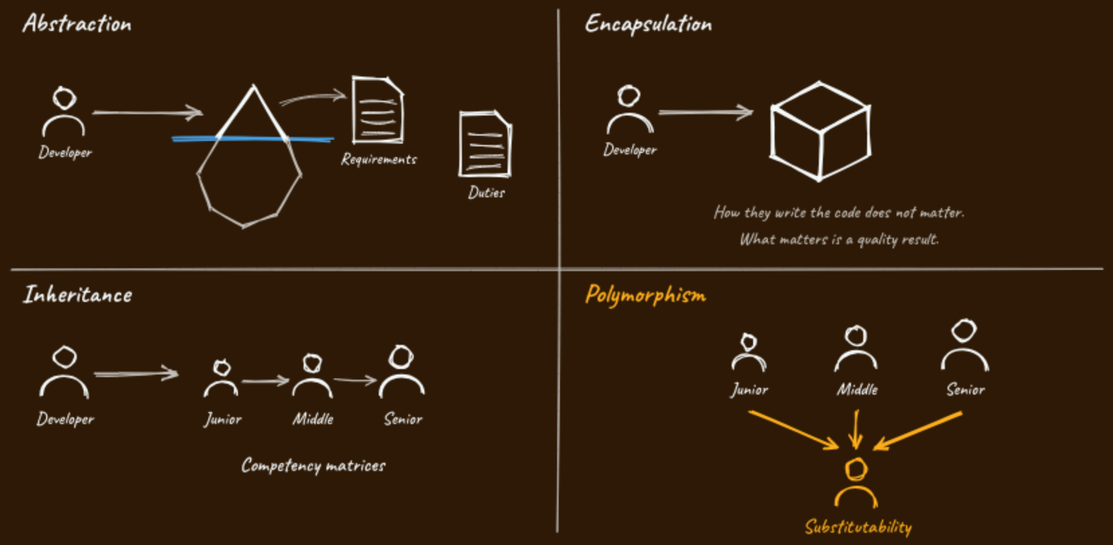


# The engineer is an object too, and is substitutable

Engineer `OOP → role`

The same four words applied to a role

- Abstraction: requirements → duties
- Encapsulation: the result matters, not the keystrokes
- Inheritance: Junior → Middle → Senior
- Polymorphism: substitutability within a level



Part 2

02

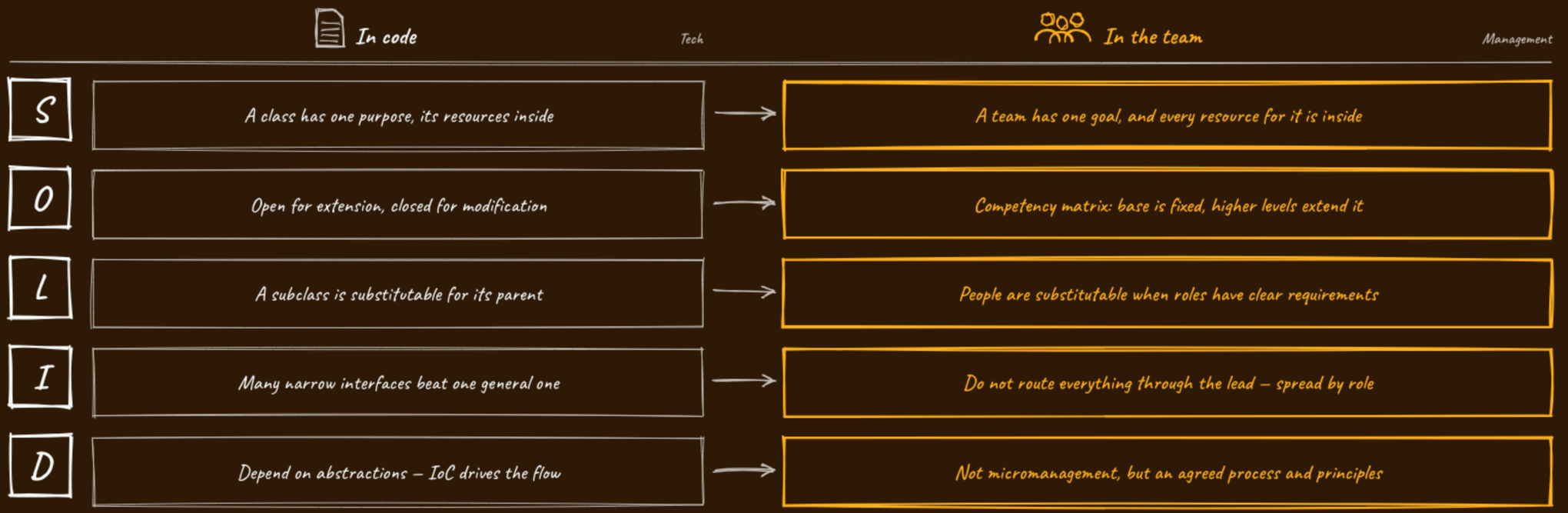
# SOLID

Five principles — and five management practices

SOLID Map

# Every principle is stated twice

First for code, then for the team

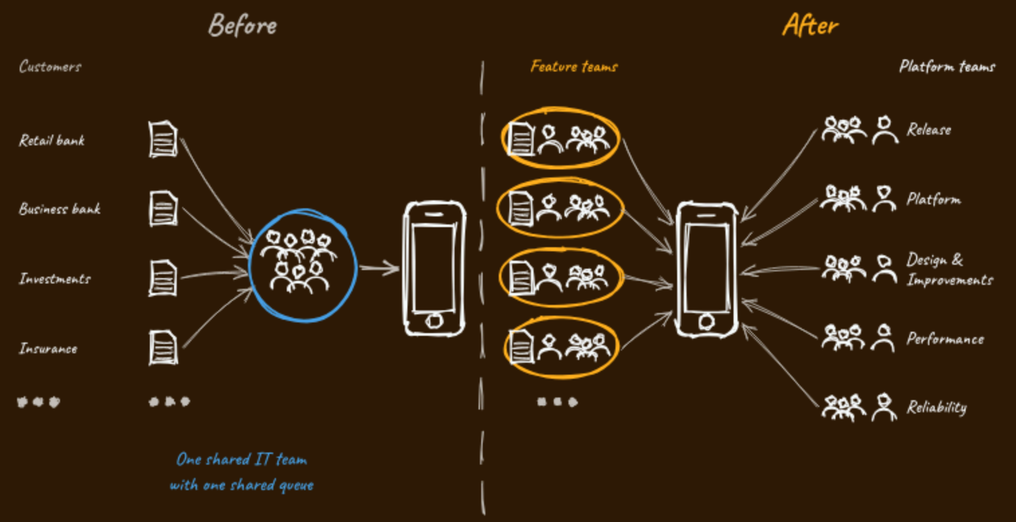


# A team has one goal and every resource for it inside

S Single Responsibility Principle

Example: from one shared IT team to feature teams

- In code: a class has a single purpose
- In a team: one goal, and its resources are inside
- Before: one shared IT team with shared prioritisation
- After: a feature team per customer plus platform teams

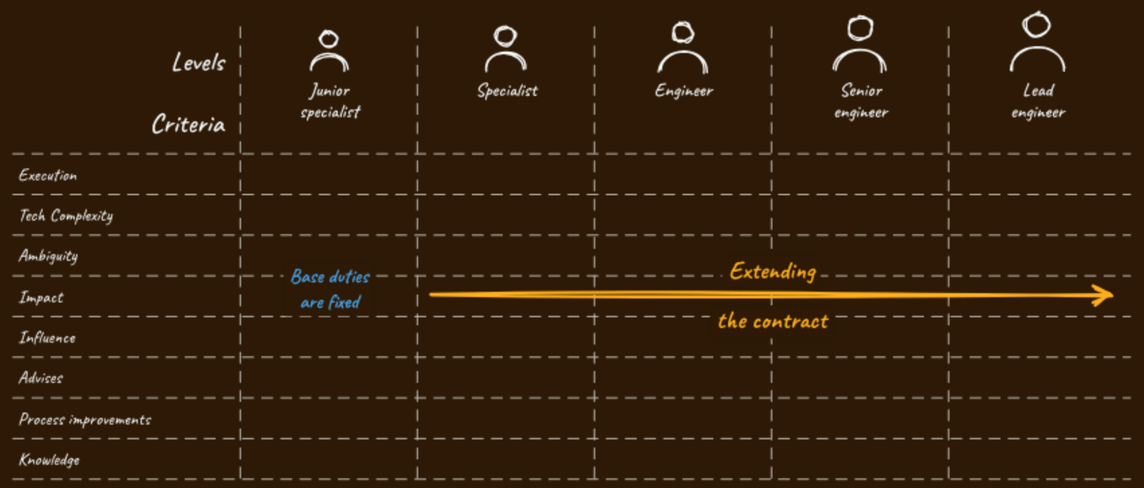


0 Open/Closed Principle

# Levels extend the contract, they do not rewrite it

Example: the QA competency matrix

- In code: open for extension, closed for modification
- In a team: this is about competency matrices
- Base duties are fixed at the junior level
- Every level above only extends the contract

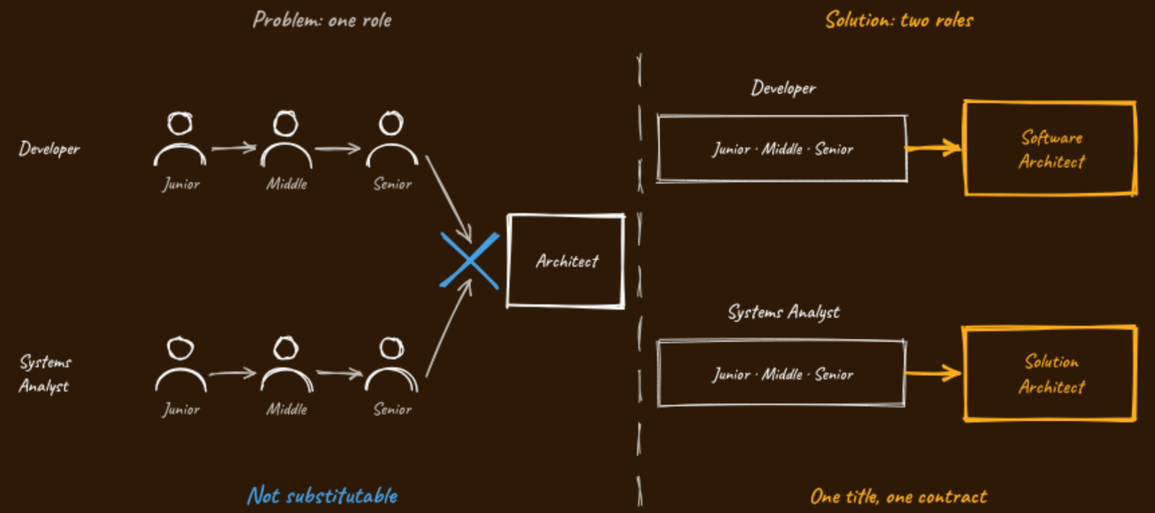


# One role title means one contract

L Liskov Substitution Principle

Example: an architect reached by two ladders

- In code: a derived class stands in for its parent
- In a team: everyone is substitutable, starting with the lead
- Problem: development and analysis both lead to Architect
- Solution: Software Architect and Solution Architect

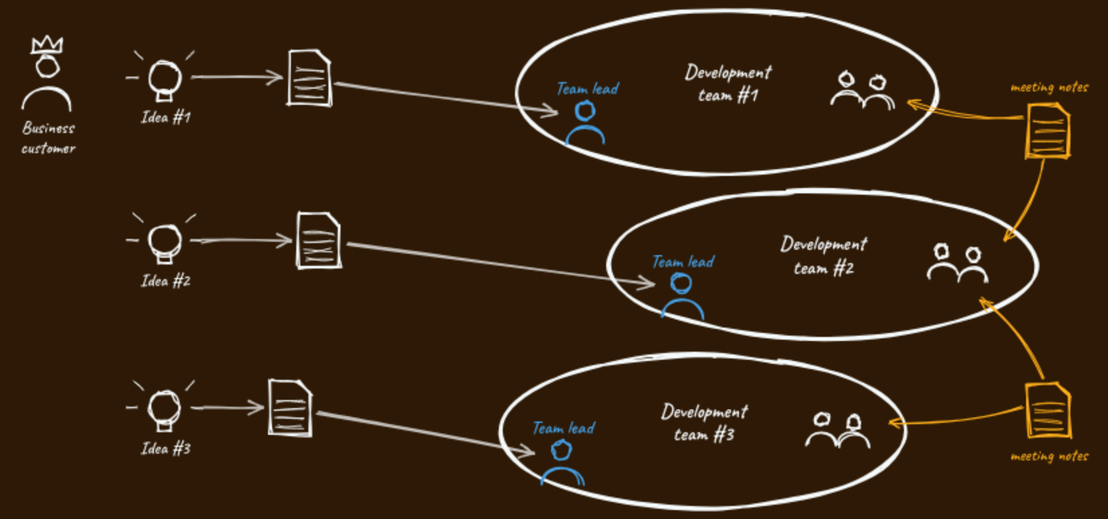


# The team lead is the interface, not the only channel

I Interface Segregation Principle

Example: segregated interfaces and meeting notes

- In code: many narrow interfaces beat one broad one
- In a team: one interface is exposed outward — the lead
- Do not route every activity through that one person
- Spread the work across the roles inside the team

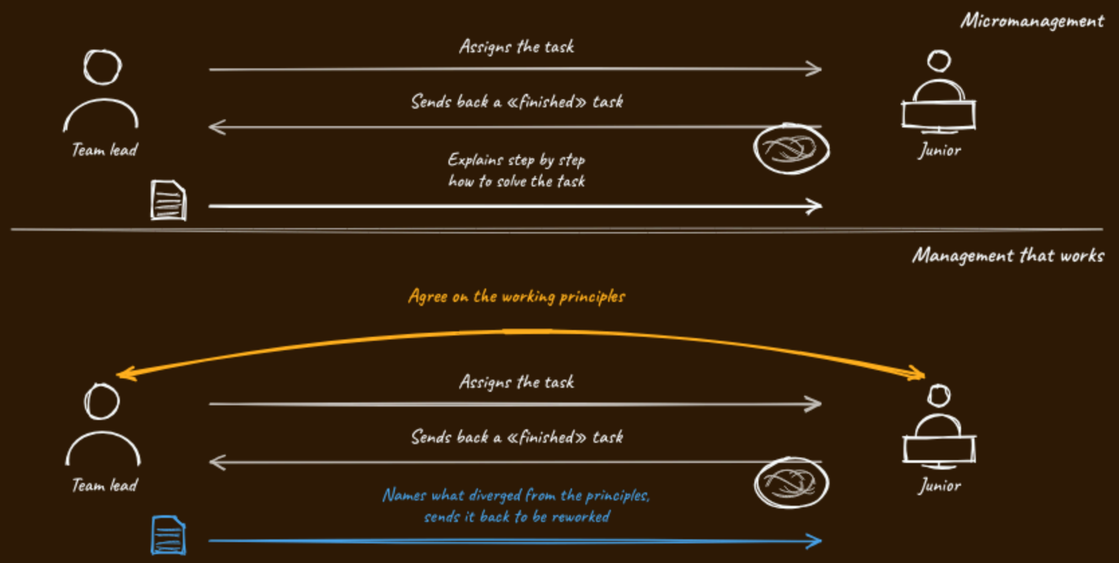


# Principles drive the flow, not the lead step by step

D Dependency Inversion Principle

Example: micromanagement vs management that works

- In code: depend on abstractions
- IoC matters more: the framework drives the flow
- Micromanagement: the lead re-explains every step
- The healthy version: principles first, then the task



Part 3

03

# Takeaways

What to take with you

# Five principles — five management decisions

The management half of SOLID

- S: a team has one goal and every resource for it inside
- O: a competency matrix fixes the base and extends the contract
- L: clear level requirements produce substitutability
- I: the team lead is the interface, not the only channel
- D: principles instead of micromanagement

- «Essentially, all models are wrong, but some are useful» — George Box

Further reading Sources

# What to read next

The knowledge sources from the talk’s closing slide

- Article «The SOLID team lead, or management basics for engineers»
- Article «SRP and IoC principles in software development management»
- Article «Orchestration and choreography in development management»
- Talks «Modern approaches to building software» and «How we make architecture decisions»

polomodov.tech

# Thank you!

All slides and links — in the Telegram channel

**Alexander Polomodov**

Technical Director & Fellow,
T-Technologies



@book_cube