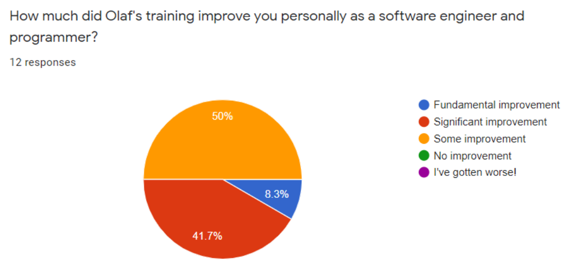
{CODE}COACH

Product Catalogue

"The organisation that needs better Software Engineering, but hasn't bought it, is already paying for it."

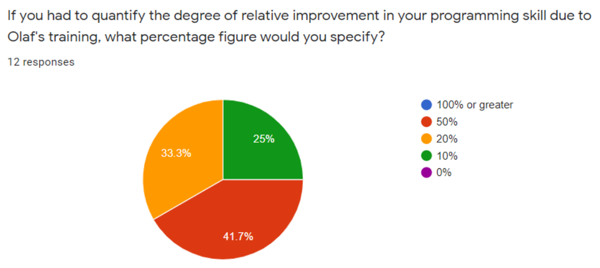
"Olaf is a master TDD practitioner and software craftsman. I highly recommend Olaf's training to anyone seeking superior software engineering outcomes."

Mark Pearl *Head of Engineering and Author of 'Code with the Wisdom of the Crowd'*



How much did Olaf's training improve you personally as a software engineer and programmer?

12 responses



If you had to quantify the degree of relative improvement in your programming skill due to Olaf's training, what percentage figure would you specify?

12 responses

Survey results from a recent 3-month client engagement: +30% improvement.

"Transcend merely learning about programming; master the art of conceptualising and strategising in programming."

Products & Services

- Talks
- Workshops
- Programming Tips

Email: olaf@codecoach.co.nz Mob: +6421430003 Website: https://codecoach.co.nz

{CODE}COACH

- [For Hire]

Notes:

- *All prices are in New Zealand Dollars and exclude 15% Goods & Services Tax (G.S.T.)*
- *Non-New Zealand purchases will be exempt from G.S.T.*
- *Auckland training sessions can also be delivered on-site. A surcharge of 20% applies for in-person training.*
- *Olaf may be available for other software engineering consulting engagements - training, advisory or otherwise. If this is of interest, please send an email to olaf@codecoach.co.nz.*

"Everything (including software!) should be made as simple as possible, but not simpler."

Talks

These presentations are delivered in a lecture format and cover a variety of topics related to software craftsmanship. After each talk, there is a Q & A session and discussion. The main goal of these talks is to introduce useful strategies for creating computer systems that are easy to maintain and comprehend. They are suitable for individuals with any level of skill, unless stated otherwise, and offer great benefits for bigger groups.

Duration: Circa 2 hours
Attendees: No limit

Prices:

- **Single talk: $2,500**
- **5-talk bundle: $10,000**
- **10-talk bundle: $15,000**

Available Talks

Please check out my list of available presentations. Click on the link to navigate to a brief description. If you are interested in a talk on a topic not covered, feel free to send me an email at olaf@codecoach.co.nz with your requirements.

Talk Descriptions

General

Talk: Why Software Craftsmanship?

Learn about the *single factor* making software development harder over time. Discover how to generate a faster time-to-market while reducing the total cost of ownership of your bespoke software systems.

Clean Code

Talk: Clean Code - An Introduction

Discover the guiding principles governing highly maintainable source code—Clean Code. Learn how to use variables and references the Clean Code-way. This talk delves into common mistakes developers make in this area.

Talk: Clean Code - Functions and Methods

How should you structure functions and methods? What is the recommended maximum length of a subroutine? How many things should it do? Welcome to the talk that will answer fundamental questions regarding function structure and behaviour.

Talk: Clean Code - Comments and Exceptions

Should we write as many explanatory comments as possible? The answer may surprise you. How much should you comment code? When is it a good idea to have a comment, and when not?
Many developers are confused by exceptions. Get clarity on exceptions—when and how to use them and how to manage them—exceptions made easy!

The SOLID Principles

Talk: The Purpose of SOLID & The Single Responsibility Principle

What are the SOLID Principles trying to achieve? At what level of software do they apply? What is the Single Responsibility Principle? The answer often surprises developers who thought they understood this principle.

Talk: The Open-Closed Principle & The Liskov Substitution Principle

The Open-Closed Principle is crucial to building modular, pluggable software systems. The Liskov Substitution Principle perplexes even seasoned software developers. This talk will explain both principles' significance in simple terms and with code examples.

Talk: The Interface Segregation Principle & The Dependency Inversion Principle

In this talk, you'll discover the genius of the Interface Segregation and Dependency Inversion Principles. Both principles apply at the class level and play a crucial role in software architecture and the design of highly flexible systems.

Object-Oriented Programming

Talk: Inheritance versus Composition

What is Inheritance? What is Composition? Inheritance is so useful, yet you probably want to use Composition in most situations. Why? If you love Inheritance but want to avoid problems, you will want to come to this illuminating session.

Components

Talk: Components & Their Effect on System Design

What are Components? Why are they so crucial to the flexibility of our systems? Constructing components with excessive coupling will cause problems in the future. Learn how to partition systems into sensible deployment units to harness the power of pluggability.

Architecture

Talk: Software Architecture & Clean Architecture

Arguably one of my best talks on how to structure software systems. Learn the answers to questions like: What is Software Architecture? Is Software Architecture still important? What is Clean Architecture? How should we begin writing software to maximise system pluggability? The target audience of this talk is more experienced, senior developers. Nonetheless, less experienced programmers will benefit from learning how to structure systems too.

Unit Testing

Talk: Writing Effective Unit Tests (NEW)

What is a unit test? How does a unit test differ from other automated tests? Arrange-Act-Assert. What does it mean to ‘unit test one thing’? How do we create great, i.e. imminently readable and maintainable, concise unit tests? How should we arrange unit tests? What are Forward-Looking Tests? How do we elegantly manage unit tests for code that connects to the database (spoiler: we don’t - we use something called a ‘Fake Collaborator’)? If there is time, a bit on TDD and/or unit testing in Legacy Code.

Test-Driven Development (TDD)

Talk: A Short Introduction to TDD

Are you interested in TDD and how this innovative and iterative process lets you write clear, concise and simple code? In this talk, you will discover the lifecycle and rules of the TDD process. You'll learn how to write good unit tests. Includes a demonstration of TDD.

Talk: Core Concepts of TDD

After recapping the principles of TDD, this talk delves into the nitty-gritty of writing sophisticated modules accessing the database and other services. This talk touches on software architecture and how to partition software systems. Learn about fake collaborators and how to use them—confused about the difference between a Stub and a Mock? No longer.

Overcoming Legacy Code

Talk: A Short Introduction to Working Safely with Legacy Code

Working with legacy code is not an exciting prospect for software engineers. Many would rather leave their jobs than work on a seriously impaired legacy codebase—and some do. It's easy to break a legacy system unexpectedly—even with minor changes. But it doesn’t have to be this way.

This important talk unveils an innovative approach to safely modifying legacy code. Consistently using this proven process will allow a team to get on top of even the worst legacy code within weeks and months.

Design Patterns

Talk: Creational Design Patterns

Learn about the importance of creational design patterns—patterns used to create one or more objects: Singleton, Monostate, Abstract Factory, Factory, Factory Method.

Talk: Structural Design Patterns

Structuring and partitioning our software well is crucial for ongoing maintainability. Such structure starts with the proper relationship between entities. Discover the different structural design patterns and when to use them: Adapter, Proxy, Facade, Humble Object, Decorator, Composition.

Talk: Behavioural Design Patterns

Behavioural design patterns affect communications and messaging between objects. Common behavioural patterns are Null Object, Strategy, Template Method, Command. This pattern category is relevant for the flexibility of program flow.

Talk: The Event Sourcing Enterprise Architecture Pattern (NEW)

What is Event Sourcing? What problems does it solve, and when would you want to use it? How does Event Sourcing differ from CQRS? What are the disadvantages of Event Sourcing? This talk includes a demo of a simple Event Sourcing system. You'll see not only the power of this pattern but also the drawbacks. Recommended.

Personal Development (NEW)

Talk: Building Personal Resilience (NEW)

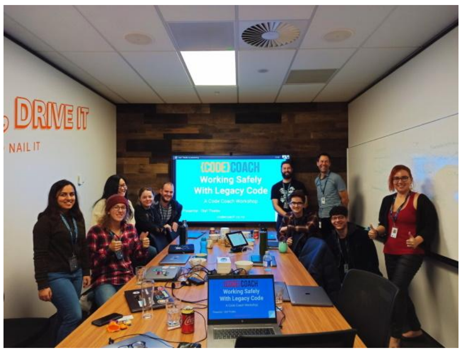
Discover how to strengthen your ability to bounce back from adversity in this engaging presentation on building resilience. Learn practical strategies and techniques to help you cope with stress, overcome obstacles, and thrive in the face of challenges. Don't miss out on this empowering opportunity to enhance your resilience and well-being!

"Software engineers are among the highest-paid people at our organisations. Why let them do their work with outdated ideas and low effectiveness? It's like driving a Porsche at walking pace—it doesn't make sense."

Workshops

Each of my workshop courses thoroughly explores a particular Software Engineering topic and features a combination of *theory*, *demo* and *hands-on lab* experience. The objective of all of our CodeCoach workshops is to provide *deep insights* and *superior long-term comprehension*.

Duration: 2-3 days (or 4-6 half-days)
Participants: Up to 16



Available Workshops

Clean Code

Workshop: Writing Clean Code

Clean Code is the first rung on the ladder to software engineering excellence. Unfortunately, writing clean and simple code is often overlooked as a means of remarkably improving code quality and program maintainability. In this workshop, you will discover that applying high-level principles to your coding activity is more beneficial than conforming to rigid rules—and you'll get the rules as well.

Even many experienced developers do not know and apply this innovative set of coding guidelines. Learn how to program well and be rewarded with simplicity in your code. Features, theory, demo and hands-on lab.

Target Audience: Novice to Competent
Duration: 2 Full days or 4 Half-days (needn't be successive days)
Participants: Up to 16
Price: $12,000

The SOLID Principles

Workshop: SOLID In Action

It is with good reason expert software engineers hold the SOLID Principles in high regard. These principles are foundational to constructing software that is easy to change, easy to understand and reusable. After this workshop, you'll know and appreciate all the SOLID Principles.

You'll learn why code in violation of SOLID is challenging to work with and why compliant code is easy to modify and maintain. If you are unfamiliar with SOLID, this workshop will add essential expertise to your software engineering toolbelt. Features theory, code demonstrations and hands-on lab.

Target Audience: Advanced Beginner to Proficient
Duration: 2 Full days or 4 Half-days (needn't be successive days)
Participants: Up to 16
Price: $12,000

Test-Driven Development (TDD)

Workshop: Complete TDD

Have you had enough of messy, hard-to-read code? Do you want to learn how to write clean, beautiful software that's easy to understand and where you can make changes quickly and effectively? This workshop includes live demonstrations of how to apply TDD to write real-world code. You'll get to use your new knowledge in a hands-on TDD programming lab.

Whether you are new to programming or an old hand, this workshop will show you an iterative programming approach that can significantly reduce code complexity compared to developing the *traditional* way. Learn about this novel technique taking the software development industry by storm! This workshop is suitable for programmers new to TDD and those with some TDD experience who would like a refresher.

Target Audience: Novice to Proficient
Duration: 2 Full days or 4 Half-days (needn't be successive days)
Participants: Up to 16
Price: $12,000

Overcoming Legacy Code

Workshop: Working Safely with Legacy Code

Legacy Code is taxing to work with at the best of times. As developers, we try to avoid making changes to Legacy Code. But sometimes we have to, and when we do, we inevitably break existing behaviour. Understandably, our customers will be angry, and our managers unhappy. What is Legacy Code? Why is working with it so demanding?

Is there a better way—a safer way—to work with Legacy Code? Yes, there is! This workshop will show you how to modify Legacy Code safely. The learning process will allow you to make the changes you want and open the door to continued safe refactoring and code improvements. This workshop includes theory, demo and hands-on lab.

Competence Level: Advanced Beginner to Proficient
Duration: 3 Full days or 6 Half-days (needn't be successive days)
Participants: Up to 16
Price: $18,000

Software Architecture

Workshop: Principles of Software Architecture

What is Software Architecture? Is it still relevant? Why? Does Software Architecture only apply at the highest system level? Where and how do software developers affect architectural design? How do we create modular and pluggable systems? Is Microservices Architecture an architecture? (Spoiler: It isn't.) This workshop answers all those and your Software Architecture questions.

Learn about Clean Architecture, a system design philosophy that turns software engineering into 'Lego for Adults'. Discover the secret to writing highly maintainable software. Learn why partitioning software using Clean Architecture while writing the code with Test-Driven Development (TDD) is a software engineering superpower!

Competence Level: Competent to Expert
Duration: 3 Full days or 6 Half-days (needn't be successive days)
Participants: Up to 16
Price: $18,000

Programming Tips

Developers can go out and actively learn how to be more productive. But are they going to do it? Human nature, being what it is, wants things to be **easy**.

Why not make learning to masterfully program **really easy** for your software engineers? Have a short programming tip email arrive in their inbox each workday. Each tip is only around 500 words and explains an important, high-value coding concept.

We have a cache of over 250 programming tips, and we are still creating more.

Improve your programmers one daily programming tip at a time!

If your software engineers absorb even 1% of the programming tips, this investment will have paid for itself!

- **RISKFREE - Try before you Buy!** This service is **FREE** for the first 10 days.
- After that **only $2 per email recipient per day**.
- Minimum of 10 developer recipients, increasing in tranches of 10 recipients.
- Bulk pricing for 100 or more recipients

Please get in touch (olaf@codecoach.co.nz) if you would like to learn more, receive sample emails, or book this service.

For Hire

This is the most intensive learning experience I offer. I teach essential software engineering principles during mob and pair programming with a team of developers on their own code.

Unlike the limited exposure to software engineering best practices during a workshop, this close-quarter training greatly increases the capacity for retention, as the material can be applied immediately in a familiar context.

Price:

- **Daily rate: $4,000**

Note: I offer a discount for longer engagements