QWAN

Quality Without A Name

Courses & Workshops 2016

Table of Contents

Introduction

We help customers to get more value from software development. We offer software development, mentoring and training under a joint label: *QWAN (Quality Without A Name)*[*]. *Quality without A Name* means that we strive for something elusive, which we can't word exactly.

We advance the state of our practice and that of our customers. To that end, we investigate new practices, apply them in the real world, and reflect. We share our experiences, good and bad, in the form of essays, blog posts, practitioner's conference appearances, courses, hands-on mentoring and simply delivering valuable software with others.

We specialize in **agile software development** (**lean**, scrum, extreme programming, test driven and behavior driven development), open space facilitation, and **systems thinking -** we see and improve the whole as well as its parts.

We offer the following services:

Contact us for:
· hands on mentoring
· open enrollment courses
· in-company training
· 1-to-1 coaching

in agile and lean development & systems thinking

www.qwan.eu

Open enrollment courses - Watch www.qwan.eu or our newsletter at www.qwan.eu/newsletter for scheduled courses. These courses can only run with sufficient demand. Let us know if there is something you're missing.

Customized in-company courses - All courses in this brochure can be run in-company, for groups of 8 or more. For smaller groups, we offer a combination of hands on coaching and mentoring, supplemented with presentations and exercises.

If you have a need that is not covered by an existing course, we are happy to make new ones for you. Our best courses got started by specific requests from our customers!

Coaching and mentoring - We help you to find areas to improve, based on the needs and actual situation of your business – one size does not fit all. If applicable, we will recommend and teach existing practices from elsewhere, but more often we will look for what is already working for you and make that better. If we recommend a change, we will make it stick by providing hands on coaching and mentoring. This works well together with courses and workshops. Workshops get everybody on the same page, mentoring resolves challenges that are specific to your context.

One-to-one coaching – We can be your sparring partner, by reflecting on your way of working we help you improve.

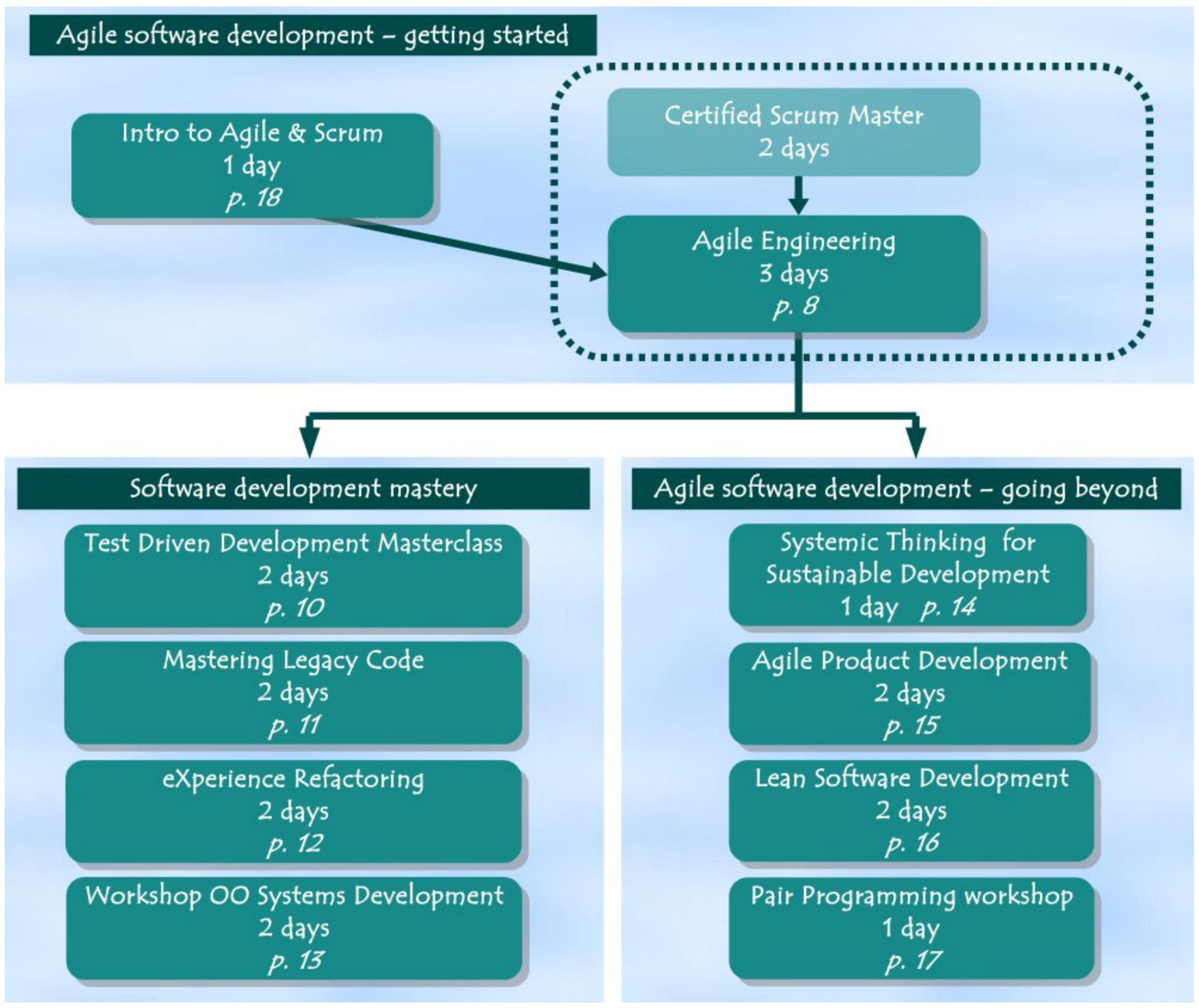
We'd love to hear your feedback about this brochure and our offerings – play the *perfection game* with us: how would you rate it on a scale from 1 to 10? Why? What can we do to make it perfect?

We look forward to seeing you and your colleagues at one of our courses or meeting you at your place of work, so that we can jointly find a way to make your work more fun and productive!

Best regards, Marc, Willem & Rob

[*] The name *Quality Without a Name* was inspired by Christopher Alexander's work on architecture and patterns. We highly recommend his books *The Timeless Way of Building* and *A Pattern Language*.

Courses overview


Agile software development – getting started
Intro to Agile & Scrum
1 day
p. 18
Certified Scrum Master
2 days
Agile Engineering
3 days
p. 8
Software development mastery
Test Driven Development Masterclass
2 days
p. 10
Mastering Legacy Code
2 days
p. 11
eXperience Refactoring
2 days
p. 12
Workshop OO Systems Development
2 days
p. 13
Agile software development – going beyond
Systemic Thinking for Sustainable Development
1 day p. 14
Agile Product Development
2 days
p. 15
Lean Software Development
2 days
p. 16
Pair Programming workshop
1 day
p. 17

Be the first to know about upcoming events and subscribe to www.qwan.eu/newsletter

All prices are exclusive of VAT

What participants have said about our courses:

"Very good and fun course. Nice introduction to a new culture of software development."

"Three very fun and educational days, Good combination of theory/presentation and hands-on exercises."

"Very educational (as I expected); I can definitely apply this in my project"

“The practices of TDD and implementation of a small project in three iterations which involved almost all practices of XP where very good for me to get real experience.”

“This was the first training where I had:
- no problem to stay awake
- no problem with the afternoon “dip”
- no problem to stick to the subject
- no problem completing the exercises.”

“Awesome! Very valuable tool to understand causal effects.”

“Well run, good materials. Clear and useful content.”

“Spent a lot of time on explanation, discussion, doing things ourselves, and less time to study materials during the course. That's good - learning by doing.”

“Learning appealing, interesting software development concepts in an accessible way: very enjoyable course!”

“Nice course - lots of practical stuff - I would recommend to everyone starting with AGILE or wanting to confront their knowledge”

Coaching and Mentoring

We offer coaching and mentoring of teams and individuals in agile development practices. Coaching and mentoring can be a highly effective follow up to a training course. We help people to improve swiftly, so they can work in new ways effectively. We also support our clients in tailoring practices to the context of their day-to-day work.

Change is a hard skill

Several things can go wrong when you try to transition to a different way of working. The scope of the change is too big, or too small, the development process becomes a boring routine, or the team doesn't go beyond 'agile by the book', so that continuous improvement never happens.

Increase your chance of success by hiring us as mentors:

- getting started as a Scrum Master
- adopting Scrum, eXtreme Programming, Kanban or Lean Software Development
- test driven development and feature/story testing (BDD)
- setting up a continuous integration environment
- architecture, design and refactoring
- iterative planning, and estimating
- leading retrospectives
- adapting a set of practices to your needs

Benefits

We have helped our customers reduce time to market, reduce defects, deliver software in difficult circumstances (limited budgets, unclear requirements), and increase the fun with which teams, their managers and their customers work.

Approach

Our approach is characterized by:

value focus - we find out where we can add most value, and leave when the return on our involvement diminishes (as your own people get better at mentoring their peers).

facilitating - we create space where people can safely experiment and learn

responsibility – everyone takes responsibility for their own learning

experiential – people learn best by doing and experiencing

systems view – let people see the whole together, to prevent sub-optimization

continuous reflection and improvement – the foundation for better performance

collaboration, process, and results focus - we take responsibility for the results of our joint effort; we can implement what we recommend and keep track of both the ball and the goal.

Courses

Agile Engineering

8-12
developers, architects, technical leads, scrum masters, project managers

3 days

€ 2150 per participant
£ 1495 per participant
in house: on request

When you have learned the basics of agile software development, you might wonder how to actually get things done and get your software out of the door when you go agile. Scrum focuses on coordination and feedback practices, adding technical practices allows teams to grow systems with confidence and safety.

This intensive, hands on course focuses on the down to earth side of working in an agile way: how to plan, build, test, and deploy the software. How to get it done using user stories, test driven development, automated testing, incremental design, relentless refactoring, pair programming, and continuous integration.

You will experience how these practices feed back into the Scrum loop and how they enhance predictability while increasing velocity.

Benefits

By participating in course, you will:

- learn how to deliver projects on-time and within budget, without compromising on quality
- see the value of eXtreme Programming principles and practices and how Scrum and eXtreme Programming are complementary
- understand how agile values, principles and practices relate
- learn how practices like test driven development, relentless refactoring, incremental design, continuous integration, and pair programming support agile principles and enable to deliver quality software incrementally and how they help to keep on delivering quality software over time
- learn how one can let quality software evolve iteratively and incrementally
- experience the role of testing and test driven development in agile development
- experience how all the different practices work together, through a real mini project

...and you'll have fun! We take pride in creating a fun and effective learning environment, mixing presentations with simulations, exercises and hands-on development.

Intended audience & prerequisites

Developers, architects, technical project leaders. Basic knowledge of agile development and Scrum is needed for this course, for example gained through a one day agile workshop or a CSM or CSPO course.

Programme

Day 1:

- eXtreme Programming – values, principles & practices
- Test Driven Development – introduction, live programming demo and hands on exercises
- Mini retrospective

Day 2:

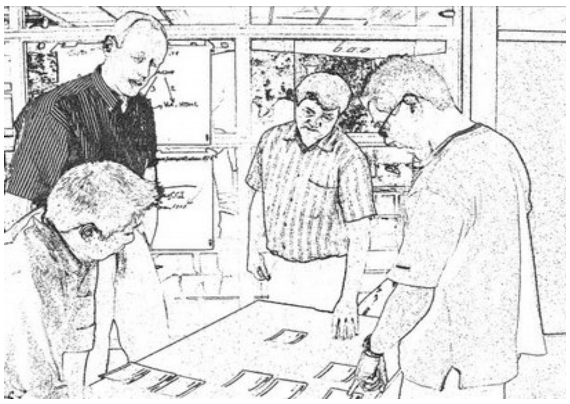
- Agile Architecture & Design
- Smells and Refactoring
- Writing User Stories
- Supporting Short Releases through Continuous Integration and Automated Acceptance Testing
- Mini retrospective

Day 3:

- Bringing it all together, in a 3 iteration micro project
- Course retrospective

We offer this course in Java, C#, and Ruby.

This Agile Engineering course is also an excellent way to prepare for the Scrum.org **Professional Scrum Developer 1** assessment. See www.scrum.org/Assessments/Professional-Scrum-Developer-Assessments



Test Driven Development Masterclass

8-14
developers, architects, technical leads, scrum masters

2 days

€ 1050 per participant
in house: on request

In this two day, in depth practical course, you will master different unit testing techniques, like mocking and unit testing in legacy code.

Benefits

When a team masters unit testing, they and their customers will benefit from a faster development cycle with fewer defects and software that has a better design. This makes it easier to adapt to changing customer requirements.

By participating in this course, you will:

- gain a better understanding of what a unit test is
- learn techniques and guidelines for writing and maintaining tests
- test code in isolation with test driven development and mocking
- experience various styles of mocking
- gain some experience adding tests to legacy code
- know how to evaluate and improve your unit tests
- understand system dynamics of unit testing

Prerequisites

This course requires knowledge of object oriented concepts, experience in a object oriented programming language (like Java, C# or Ruby), and basic knowledge of automated testing. Experience working on software development projects is also useful.

Topics

The course consists of a mix of short presentations, live programming demonstrations, and lots of programming exercises.

- Why Unit Testing? - short and long term systemic effects
- Test Driven Development
- Taking Test Driven Development to the extreme
- Responsibility Driven Design with mocking
- Interaction based vs. state based testing
- Mocking Styles – when to use mocks, stubs, fakes, and dummies
- Getting Your Tests In: Breaking Dependencies in Code
- Unit Testing Clinic

We run this course in three different versions: Java, C#, and Ruby.

For the Java and Ruby versions of this course, we will provide laptops with a complete development environment. For the C# version of this course, you need to bring a laptop with Visual Studio (2005 or newer) with NUnit installed.

We can adapt the program if desired when we run the course in-house.

Mastering Legacy Code

8-14
developers, architects, technical leads, scrum masters

2 days

€ 1050 per participant
in house: on request

In this two day training course, you will learn how to improve the design of software step by step, while continuing to deliver value to your customers.

Everyone wants software that 'just works', that is a joy to maintain, and where you can add new features easily. In practice however, there are always bugs to fix and customers to please. Corners are cut and design debt accumulates, ultimately resulting in *legacy code* – code without tests.

You would like to improve the software in small steps using *refactoring* - which means improving the design of existing code without changing its behaviour. There is a Catch-22 here: to refactor safely, you need automated tests to ensure the code behaves the same as before. To add tests, you first have to make it modular by refactoring.

There still is hope! Through presentations, demonstrations, and exercises, you will learn to identify smells, break dependencies, refactor, and add tests in messy legacy code, so that you can add new features with more ease and confidence. This course helps you get started and offers effective practices and courage to persevere.

Benefits

If you participate in this course, you will:

- learn a number of techniques to break dependencies in code
- know how to start making changes and adding tests responsibly
- be able to find *seams* in code and use them to break dependencies
- understand the system dynamics of design debt
- experience prioritizing, planning, and executing a large change in small steps with a team, on real, representative legacy code.

Prerequisites

This course requires knowledge of object oriented concepts, experience in a object oriented programming language '(like Java, C# or Ruby), and basic knowledge of automated testing.

Topics

The first day is introductory, with an overview of refactorings, code smells, and how to break dependencies by finding seams in the code. We will do this through a mix of presentations, demonstrations and small exercises. You will also learn about the systemic causes and effects of design debt and how refactoring in small steps can help you break out of a vicious cycle.

Day two is the hands-on day. In two parts we will plan and execute a large refactoring on a real-world legacy system. Participants are coached in applying what they have learned: working on a large, messy legacy code base without getting lost, using the vocabulary of smells, refactorings, seams, and dependencies to co-ordinate, plan and execute a large refactoring with a team. And do so safely, even though in the beginning no automated tests are present.

eXperience Refactoring

8-14
developers, architects, technical leads, scrum masters

2 days

€ 1050 per participant
in house: on request

Program code goes in maintenance after the first line has been written. Every modification after that first line is a change to existing code. Badly written code is usually not written by bad programmers, but by good programmers doing their best. They design an elegant solution for a problem, but once the initial solution is out there, the code doesn't get enough attention. So when it gets modified it will degrade over time.

This course is a practical, hands on introduction to refactoring – improving the design of software without changing its external behaviour. Refactoring is a practice to keep your software in good shape, by continuously performing small, well-defined, safe improvements. Refactoring is not changing everything and breaking all the tests, or doing a grand redesign. Refactoring is taking small, safe, steady steps to a better code base.

Benefits

If you participate in this course, you will:

- be able to improve design in code in small steps responsibly
- know a number of refactorings and how to apply them
- recognize code smells and know what refactorings to apply
- be able to do small refactorings on the fly, continuously improving a design bit by bit
- apply refactoring as an essential part of test driven development
- know how design patterns and refactoring relate
- be able to refactor to and from some design patterns

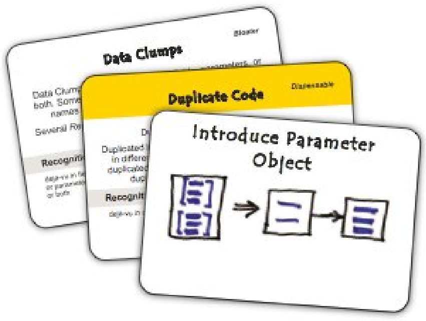
Prerequisites

This course requires knowledge of object oriented concepts and experience in a object oriented programming language '(like Java, C# or Ruby). Experience working on development projects is useful as well.

Topics

This course is a mix of bits of theory and lots of practice. We will cover:

- What is refactoring?
- Smells & refactorings
- Refactoring and test driven development
- Giving your code some love
- Refactoring to & from patterns
- Large refactorings



Workshop OO Systems Development

8-14
developers, architects, technical leads, scrum masters

2 days

€ 1050 per participant
in house: on request

Join us for two days of fun games and serious exercises. Take Object Oriented design to the next level in this highly experiential course. We'll practice OO Design and programming and show how unit tests help to drive your design.

You will get plenty of opportunity to practice practical design techniques like Class Responsibility Collaboration cards, test driven development, refactoring and mock objects with design and programming exercises.

You will learn to apply good OO design principles and to recognise bad ones. You will experience tackling these step by step. In other words, you learn to care for your code in a structured way so that it becomes easier and more fun to maintain and extend.

This course is available in C#, Java, and Ruby; other programming languages are possible on request.

Benefits

You will improve your design skills through hands on practice and reflection. You learn a low-ceremony way to create and implement a design that is simple, robust and understandable. This means you can deliver software quickly, while at the same time reducing maintenance costs.

You will learn how to improve the design of legacy code so that working with legacy software becomes more predictable (and fun!) and work on the software has a higher return on investment.

You will also learn what causes technical debt in your place of work. You will practice systems thinking techniques to understand what makes it hard to apply good design practices, and find out how your team can reduce and preventing technical debt.

Prerequisites

Basic knowledge of object orientation programming concepts and automated unit testing is required.

Topics

- Class Responsibility Collaboration cards, the simplest design technique that could possibly work
- SOLID design principles
- Using test driven development to drive good design
- Evolving OO design with refactoring
- End-to-end test driven development
- Responsibility driven design using mock objects
- Seeing system dynamics of (lack of) refactoring, to understand what makes refactoring hard
- OO & legacy code - getting your foot in the door with refactoring and unit tests

Systemic Thinking for Sustainable Delivery

6-20
managers, project leaders, IT customers, scrum masters, coaches

1 day

€ 525 per participant
in house: on request

In our work, we have noticed that changing your software development method changes your organisation. We have run into questions like:

- Why can't the customer keep up with the development team and what can be done about it?
- Why is it so hard to introduce agile in some organizations?
- What are common failure modes of agile transitions and what can we do to work around these?
- How can we get a team to start unit testing (or any other good technical practice)? Is lack of unit testing really the biggest problem?
- What are the underlying dynamics of adding people to a project and how can we use these to increase project velocity?

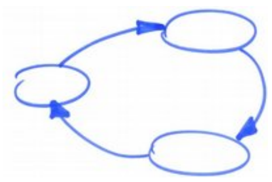
While several development methodologies describe a recipe for practices that worked in a specific context, the exact same recipe is not guaranteed to work in all circumstances.

Along our journey we have discovered and mastered a number of useful thinking and doing tools that help us become better reflective practitioners and find answers to the above questions, tools that we have found useful in making sense of what's happening in our organizations.

Benefits

This course helps you to effect substantial change in small, controllable steps, and prevent local (sub)optimization. Understanding why your process works the way it works helps you see the extent to which a practice is effective in its context. It also helps your organization grow its own way of developing software.

After participating in this course you will:



- view software organizations from a different, systemic perspective
- find root causes for all kinds of persistent, messy team problems
- understand why different software organization work and can work and why they sometimes stop working effectively
- adapt your change initiatives to the specific organizational context
- understand how change happens in individuals and organizations, and what this means for steering a change process

Prerequisites

We expect participants to have experience working with software projects, as a manager, customer, developer, or other stakeholder.

Topics

- Introduction to systemic thinking, with hands-on exercises using issues the participants bring in
- Value stream mapping
- Cynefin sensemaking framework
- Cultural patterns of software development organizations

Agile Product Development

6-20
IT managers, IT customers, project managers, architects, team leads, scrum masters

2 days

€ 1050 per participant
in house: on request

Once their development teams have embraced agile development and know how to deliver working software frequently, many organizations find that the bottleneck no longer is within engineering. It shifts to the realm of product management: *how to do effective product development*?

Scrum states that the Product Owner should create and maintain a prioritized product backlog, eXtreme Programming requires the Customer to speak with one voice. In practice, this is easier said than done. The customer realm is a complex, but essential, part of the value stream. Striving for excellence in engineering and project management practices is not enough, mastering agile product development is also necessary.

Benefits

In this two day intensive, practical course, you will learn to:

- Prioritize by business value
- Understand the concept of business value
- Write good user stories
- Make sure a story is really done and stays that way
- Apply alternative agile planning strategies
- Manage many stakeholders with conflicting requirements
- Manage stakeholder expectations for incremental delivery

Prerequisites

We expect participants to have experience working in agile projects and background knowledge of agile principles and practices.

Topics

We offer tools and techniques for release planning and management, prioritizing, effective analysis, story writing, and story testing. We will also cover strategies like User Story Mapping, Kanban, and Dimensional Planning which help you deliver business value to multiple stakeholders in a sustainable way.

Lean Software Development

6-20
IT managers, IT customers, project managers, architects, team leads, scrum masters

2 days

€ 1050 per participant
in house: on request

Competitive advantage through continuous reflection and improvement

Lean is a way of thinking that originates from manufacturing and production processes. Thanks to the sustained success of companies like Toyota and Zara, interest for lean is growing.

Lean thinking helps you to deliver increasingly better products faster. The principles of lean can be summarized as:

- respect for people
- relentless elimination of waste from processes

Waste is every activity that doesn't directly add value for the customer. Lean means continuous reflection and continuous improvement. It also means seeing people as the key factor and continuously developing people.

In this course, you will learn principles and practices of lean software development. You will also practice with a number of concrete conceptual tools, so that you can apply these in your daily work.

Benefits

After participating in this course, you will be able to:

- explain what the principles and background of lean software development are and how lean relates to other agile processes
- determine where and when lean thinking and lean practices add most value within your organisation
- apply a number of lean tools and use them to run projects (and your organization as a whole) more effectively

Prerequisites

We expect participants to have experience working with software projects, as a manager, customer, developer, or other stakeholder.

Topics

- Principles and background of lean software development
- Systemic approach to (IT) organizations - local vs. global optimization; single, double, triple loop learning
- Theory of Constraints
- Effect of measurement systems in organizations
- How lean relates to (agile) software development
- Set based development
- Identify and optimize the whole using value stream mapping
- Kanban, push vs. pull & one piece flow
- Role of testing and reviews
- Introducing lean in an organization

This interactive course consists of presentations, assignments, exercises, and simulations. There will be ample room for questions and discussions.

Pair Programming Workshop

6-20
software developers, architects, testers, scrum masters, agile coaches

1 day

€ 525 per participant
in house: on request

Pair programming is an agile practice that is sometimes seen as controversial. It seems inefficient and hard to sell to management. However, it turns out to have all kinds of favorable (side) effects, not just better software quality but also more knowledge sharing, joy of work and faster development.

What is the underlying dynamic of pair programming? What are the conditions for it to work? How can you do it effectively and enjoyable? In this workshop, we dive into the how, what & why of pair programming, with a bit of theory and lots of exercises and simulations.

Audience

This workshop is for software developers, architects, testers, scrum masters, agile coaches. A bit of programming background is needed for some of the exercises

Topics

Pair programming in context - understand what pair programming is and where it comes from.

Experience communication through pairing - understand how pair programming contributes to knowledge sharing and can replace part of traditional documentation.

Experience 'selflessness' - experience the importance of communication, practice thought sharing.

To pair or not to pair - understand how situational factors can influence pair programming negatively and what you can do about it.

Introduction to agile development & Scrum

6-30
developers, analysts, architects, testers, team leads, project managers, IT managers, IT consultants

1 or 2 days

on request

Everybody seems to be doing or being 'agile', everyone in his own way. What is it all about? Scrum is hot, XP is not, or is it? What exactly is 'agile'? Which methods and processes are there? What does agile mean in an broader organizational context? Is agile the ultimate goal or is there more?

In this workshop, we provide an overview of agile software development, from underlying values and principles to the concrete, well known practices and processes. We will show the role agile can play for developing organizations. You will experience an iterative process and a real retrospective to round up the workshop.

Benefits

After participating in this workshop, you will:

- have an overview of what agile software development is about
- know the most important methods that fall under the agile umbrella
- see agile development and agile transitions in a broader context
- have experienced planning iteratively and holding a retrospective

Prerequisites

None

Topics

- Introduction to agile: manifesto, values, principles
- Benefits, success stories, failure modes
- Overview of Scrum, eXtreme Programming, and Lean Software Development
- Simulation of iterative planning
- Cultural patterns of software development organizations
- Retrospective - an essential practice to establish a culture of continuous reflection and improvement.

Workshops

Workshops

6-20
developers, analysts, architects, testers, team leads, project managers, managers

½ day or 1 day

in house: on request

We offer a number of workshops that can be run as a half day or a full day in-house session. The hands on programming workshops are available in C#, Java, and Ruby; workshops in other programming languages are available on request. We can also do these hands on workshops using your team's code base as a case.

Contact us for more information: together we can design a program that fits the issues and needs of your organization.

Introduction to Test Driven Development

In this workshop, you will learn what test driven development is, how it works, and why it works, through a live programming demo and a hands on exercise. We will cover the *red-green-refactor* rhythm and the principle of working in baby steps.

User Story Mapping & Dimensional Planning

User Story Mapping is a pragmatic technique to build and maintain a product backlog and to identify & prioritize valuable user stories. It helps to quickly define meaningful releases and to provide a frame of reference to verify the business value of user stories.

By creating a two dimensional map of the product instead of a linear backlog, User Story Mapping helps to see both the forest and the trees.

User Story Mapping works well with Dimensional Planning, a compatible approach to quickly and accurately create an iterative release plan that also communicates well to clients and other stakeholders.

These are valuable tools for product owners, clients, and teams to get more grip on product development and build products that add real value.

Writing User Stories

In this workshop, you will practice story writing, learn techniques and heuristics for formulating, splitting, and refactoring user stories, so that you can make user stories that are easy to understand, estimate and build.

Acceptance Test Driven Development

As a developer,
I want to know story testing frameworks and practice one of them,
so that I can test stories using examples the product owner writes together with me

As a product owner,
I want understand the process of testing stories and practice specifying acceptance criteria using example scenarios,
so that I can help developers develop what I want

In this workshop, you will learn and practice writing automated tests for user stories, based on examples, using a story testing framework like Cucumber or JBehave.

Promise is Debt

Most teams run into technical debt sooner or later - suboptimal code, design, and tests accumulate over time. The team slows down more and more. It's a dilemma: do you tackle the issues right now? That will delay the feature or defect you're working on. Create a workaround? Then you can finish your task, but in the long term, it will only make things worse...

In this workshop, you will get insight into the underlying dynamics of technical debt and you will learn ways to break the vicious cycles.

Cultural Patterns of Software Organisations

If you want to see software teams and organisations differently, from a people and culture perspective, attend this workshop! You can reinvent the wheel, but you don't have to - therefore we based this on session on Gerald Weinberg's cultural patterns.

There are many models and frameworks that focus on processes. They're about maturity, about what one ought to be doing in order to be effective. In this workshop, we'll present a fresh perspective on software organisations, a people oriented instead of process oriented view. The model focus on subculture and people's behaviour (interactions) and will be a useful extension of your personal toolbox.

Dirty Jobs

When you start unit testing on an existing project, you are often confronted with code that is initially difficult to test. Dependencies are poorly managed and even writing the first unit test seems a sheer impossible task. Where and how do you start? Why is it your dirty job? Can you still leave before it is too late? Can anybody help?

In this hands on lab we will define the legacy problem -code without tests- and discuss the risks of making changes in such a code base. Then you can practice on real dirty legacy code, where we will guide you in some techniques to add features safely, avoiding pitfalls like task-scope creep.

Rightsizing your unit tests

This workshop helps programmers who have started unit testing to grasp the consequences of advanced testing techniques, and managers to understand the short term and long term consequences of trade-offs made by developers. You will examine several code examples to learn how to answer questions such as: what is the right size or scope for a unit test? How does the design of production code influence tests and vice versa?

Code Smells & Refactoring

This workshop gives a practical introduction to refactoring and code smells. We will present refactoring patterns and an interactive exercise on refactoring. The exercise consists of applying refactoring in small steps to a common code base, and will involve real programming.

Give your code some love!

Program code goes in maintenance after the first line has been written. Every modification after that first line is a change to existing code. Badly written code is usually not written by bad programmers, but by good programmers doing their best. They design an elegant solution for a problem, but once the initial solution is out there, the code doesn't get enough attention. So when it gets modified it will degrade over time.

In this workshop, we will focus on a number of code fragments that seem alright at first sight, but that can still be improved - they just need a little love. Just a bit of attention can make a huge difference.

Responsibility Driven Design with Mock Objects

Responsibility Driven Design is a concept coined by Rebecca Wirfs-Brock and is about good object oriented design. OO design is about assigning the right responsibilities to the right objects and developing a clear design with loose coupling and high cohesion.

Test Driven Development (TDD) guides you in that direction, but not far enough. Testing with mock objects takes it a step further. The focus shifts from state to interactions - the messages objects send to each other. Testing with mocks emphasizes behaviour and responsibilities.

In this workshop, we will show how using mock objects helps to develop in a responsibility-based instead of a state-based way. We will also show how this approach enforces the *Tell, Don't Ask* design principle.

Agile Politics

If you believe corporate politics is something for 'those dirty managers' think again. Everybody behaves in a political way when a limited amount of resources has to be divided over groups of people. In a game you will represent a political faction lobbying your peers to spend money for your cause.

This game will show you how political many common project decisions are. Whether you care about corporate politics or not, they have an impact on your work life. Whether you are a leader, a reckless reformer or "just a programmer" this workshop is for you!

Trainers

Rob Westgeest

After years of experience with Object Oriented Software Development with UML, several development processes and project approaches as developer, trainer and project leader, Rob worked on his first XP project in 2000. And with great success! He supports projects and people in the application of agile practices, principles and values since then. Rob develops himself and others continuously by visiting, organising and hosting workshops at conferences and user group meetings like SPA, XP Days, XP-NL and Agile Open.

Rob explains hard problems in a simple way, so the problems and the solutions are easy to fathom. He is able to let others experience what he learned quickly, and so doing guides teams around pitfalls. His enthusiasm and sense of humor makes it a great pleasure to work with Rob.

Willem van den Ende

Willem van den Ende is a Dutch eXtreme Programming pioneer. Since 1999 he guides organisations in the introduction of Agile Software development as an all-hands person: coach, developer and facilitator. Always active in the local and international community, he has served as board member of the Agile Alliance, host of systemsthinking.net and the European Agile Open conferences. Willem is an appreciated workshop facilitator at practitioners' conferences like XP(Day), Software Practice Advancement and Agile200*.

Willem's sharp vision, his broad knowledge, and twenty five years of experience as programmer and coach enable him to adopt a very flexible and improvising attitude during workshops. He has the ability to let people see things differently.

Marc Evers

Marc works as an independent coach, trainer and consultant in the field of (agile) software development and software processes. Marc develops true learning organizations that focus on continuous reflection and improvement: *apply, inspect, adapt*.

Marc also organizes workshops and conferences on agile and lean software development, extreme programming, systems thinking, theory of constraints, and effective communication. Marc is co-founder of the Agile Open and XP Days Benelux conferences, and the Agile Holland user group.

Marc knows how to combine his real-world experience with knowledge that is out there to create novel solutions. He likes to add games to his highly-rated workshops, so participants have fun and learn from experience.

QWAN

Quality Without A Name

Contact: [marc / rob / willem]@qwan.eu
Phone: (+31)(0)85 877 9458
Newsletter: www.qwan.eu/newsletter

The QWAN partners are:

Willem van den Ende
Living Software
livingsoftware.co.uk
+44 743 865 1672

Rob Westgeest
Westgeest Consultancy
ww.westgeest-consultancy.com
+31 6 4577 6328

Marc Evers
Piecemeal Growth
www.piecemealgrowth.nl
+31 6 4455 0003

Edition: May 2016