A software craftsmanship training programme for senior developers

Prospectus

Summary

The XP Surgery is pleased to offer a unique and bespoke mentoring and training programme for senior software developers that consists of a highly effective mix of classroom training, solo and group study exercises, and one-to-one on-the-job mentoring.

The programme covers a wide range of software craftsmanship skills, and has been run successfully with numerous software teams throughout the UK, including with the BBC's Future Media and Mobile Platforms group. The programme can accommodate between 6 and 10 senior developers.

We provide a range of course modules addressing topics as diverse as release planning, unit testing and problem solving. Then you structure the programme by selecting the topics you need most – and this includes the option to suggest topics that aren't currently on our menu.

Training workshops and mentoring sessions take place on your premises, using your software development tools. This makes the programme more convenient for your developers and more relevant to their daily work.

“I learned more in that one day of test-driven development training with Kevin than in all of the other courses I've done combined.”

Senior software developer, laterooms.com

Programme structure

Our experience has shown that participants in this programme need plenty of time to practice and discuss the ideas learned. We therefore structure the programme around a fortnightly schedule.

Each fortnight, the participants will spend one morning in a training workshop covering one of the programme's topics; our trainer will then spend the afternoon of that day working 1-1 with one of the participants. All participants will also be given “homework”, to be done in time for the next workshop:

Day 1: morning	Discussion of last week's homework New topic training workshop
Day 1: afternoon	1-1 on-the-job mentoring with one participant
	Learning review
Days 2-14	Homework, and embedding new practices into the team's day-to-day working

Due to the amount of group working and round-table discussion involved, the programme can cater for only 6-10 participants. If more developers are to be included in the training, it would be best to schedule two programmes, to run either concurrently or consecutively.

Note that the morning training workshops will involve all participants, and the one-to-one afternoon mentoring sessions will each involve a single participant, rotating through them on an equal basis.

The programme's fortnightly schedule gives participants plenty of time for homework and to embed the learning into their daily practices.

The participants

The programme can accommodate between 6 and 10 senior developers. The majority of workshop exercises will be conducted in pairs, so it is highly beneficial to the participants if they are even in number. Thus the course should be operated with 6, 8 or 10 developers.

Most of the training workshops will assume a fair amount of software development experience. They are intended for developers who have worked in application development for a few years, usually in C# or Java-like languages, and who wish to take their craft to the next level and become thought leaders in your organisation.

Exercises that involve programming will be set mostly in the language you use daily – although a few may ask participants to explore how some idea translates into an unfamiliar language or technology.

Previous course attendees have noted that much of the value of our training methods comes from the combination of group discussion and embedded practice. To that end we ask that all participants do a set of “homework” exercises each week. These will involve a variety of research, programming, facilitating team discussions or exercises. It is essential for the success of the course that you make available the necessary 2-3 hours per week so that the participants can complete these activities without feeling they need to use personal time.

Each participant will need to have an approved commitment to:

1. Half a day of workshop training each week.
2. Half a day of one-to-one mentoring during the programme, on a rotation basis.
3. 2-3 hours per week of homework, comprising a mix of solo and group study exercises.

Logistics

The course takes place on your premises, using your software development tools. This makes the programme more convenient for your developers and more relevant to their daily work.

Therefore we ask you to provide a training room for each morning, away from the distractions of day-to-day work, and including at least a whiteboard and enough software development machines for the group, each hosting a good development environment and all of the developers' habitual tools. The room should be large enough to accommodate eleven people seated comfortably at a table or tables, with room to work.

We will provide each participant with a programme workbook (in an A4 ring binder). Each week we will add new topic materials to this, and the participants will also use it to collect their working, course notes and homework.

Programme kick-off

In order to design a set of appropriate content for your developers, we prefer to kick off the programme by conducting a 2-day "deep dive" into your code and processes. During these two days our consultant will spend as much time as possible with the intended participants: pairing on different codebases, demonstrating some of the techniques from the training course as appropriate, and attending whatever agile ceremonies occur during those two days (daily standups in particular).

At the end of this visit we will sit down with the programme participants to discuss content options for the programme. At this meeting, the participants will also be given their first homework task, which will usually be a research project that must be conducted as a group.

During the next 2-3 weeks, prior to the start of the course proper, our consultant will propose a set of modules for your specific instance of the programme.

As we are all agile practitioners, it is not unusual to find that the content of later parts of the programme evolves once we get into it. Given sufficient notice to prepare course materials, we are always happy to amend the programme plan, as we all learn how the early modules have influenced the participants and their teams.

Course modules

The training part of the programme consists of as many modules as you need. During recent years we have delivered modules on a diverse range of topics in software craftsmanship, including:

Systems thinking and the Theory of Constraints	*The Boy Scout rule*
Continuous delivery	*Primitive obsession*
Story mapping	*Black box testing*
Nested feedback loops	*Composition over inheritance*
Slicing projects to deliver value earlier	*Modelling techniques*
The walking skeleton	*Effective pair programming*
Outside-in development	*Effective mob programming*
Effective bug-fixing	*Refactoring tools*
Testing untested code	*Design patterns*
Writing good unit tests	*What is habitable code?*
Extracting independent responsibilities from tangled code	*Whole values*
Understanding the Method Object pattern	*Telling, not asking*
Planning large refactorings	*The 4 rules of Simple Design*
Creating tests that communicate intent	*Modern Extreme Programming*
Injecting code to reduce coupling	*Refactoring legacy code*
Simplifying code using data structures	*Safe refactoring mechanics*
Immutable objects	*Structured problem solving*

Architecting for testability	*Facilitating design discussions*
Understand the open-closed principle	*Finding good names*
Anti-corruption layers	*Event sourcing*
Dependency inversion	*Measuring the code*
Test driven development	*Continuous improvement*

Note that many of these topics will require two half-day workshops.

The programme is not limited to the above topics. If you have a particular topic that your developers need to cover, we will be happy to add a bespoke module for you.

The ideal length of the programme varies from team to team. Our experience suggests that 8-12 modules provides a good, solid first pass. We then often find that a further round of 4-8 modules may be desirable a few months later, after the first round of learning has had an opportunity to embed itself within your practices.

Sample programme outlines

To give a flavour of the content of a typical programme, listed below are outlines of a sample of the programmes we have run recently.

Menu A:

This programme suited an organisation with a very large legacy codebase and senior developers who had little experience of modern programming techniques.

1 *Slicing projects thinly I*

2 *Slicing projects thinly II*

3 *Testing legacy code*

4 *Refactoring legacy code I*

5 *Refactoring legacy code II*

6 *Incremental development*

7 *Planning large refactorings*

8 *Writing good unit tests*

9 *Eliminating conditionals*

10 *Effective use of Resharper*

11 *Effective pair programming*

12 *Ports & adapters architecture*

Menu B:

This programme suited an organisation whose development processes had been established during the 1990s.

1 *Habitability and coupling*

2 *TDD as a Design Technique*

3 *Trunk-based development*

4 *Writing good unit tests*

5 *Incremental development*

6 *Effective pair programming*

7 *Testing legacy code*

8 *Refactoring legacy code*

9 *Composition over inheritance*

10 *Ports & adapters architecture*

Menu C:

This programme suited a young organisation with very immature project inception and governance.

1 *Habitability and coupling*

2 *Continuous improvement*

3 *Slicing projects thinly*

4 *Incremental development*

5 *TDD as a Design Technique*

6 *Composition over inheritance*

7 *Effective pair programming*

8 *Modelling*

Mentoring approach

Ideally the afternoon spent with each of the participants would involve a representative mix of their usual responsibilities. The mentor will attend all meetings and tasks with the developer, intervening with advice and guidance whenever the developer is seen to need a role model.

Interventions will focus on “craftsmanlike” behaviours such as:

- finding ways to deliver value early;
- thinking before coding;
- using the appropriate tools for the job;
- automating error-prone activities;
- seeking to avoid rework by focusing on learning outcomes;
- creating opportunities for reflection with colleagues;
- ensuring shared understanding of designs and planning decisions;
- demonstrating trust in the abilities of others;
- seeking opportunities to eliminate the root causes of problems.

In order to embed learning, the last half-hour of each day will be set aside for a one-to-one review. This will allow time for the mentor and the developer to reflect on the afternoon's events and identify personal improvement actions and activities. Should the programme allow for two or more mentoring sessions, subsequent afternoons will begin with a half-hour review of these improvement actions.

About us

Our principal coach, Dr Kevin Rutherford has been programming since 1975 in over 20 languages. He has also been a team leader, business analyst, project manager and business owner. He was an early signatory to the Agile Manifesto, a speaker at the very first Extreme Programming conference, and founder of both AgileNorth and XP Manchester. His book R*efactoring in Ruby* deals with how to identify and fix code smells in dynamic languages, and his blog http://silkandspinach.net covers a wide range of topics in software craftsmanship.

This software craftsmanship programme draws on this experience and packages it in a unique way. It has been run numerous times since 2014 with software teams across the UK, including with the BBC's Future Media and Mobile Platforms team.

To date the programme has trained over 150 developers; of these, over 95% would recommend it to others.