# Manisfesto
# How to be a happy RMoD Pharoer

## Introduction

As a team it is important to have common working culture and to follow the same rules. This contributes to a good working environment and slow the learning curve for newcomers. It also reduces the misunderstandings and all in all leads to a more productive and happy team. This manifesto will introduce the rules used among the RMoD team.

Take these rules as the common denominator shared among all team members.

## 1 Be a Team Player

The team is not just a set of people sitting by accident in the same building. Being a team means taking benefits of the interaction between its members. This interaction can be social or more work oriented.

### 1.1 Meet the People

A team is basically composed of humans. Social interactions are important to know each other. When you have to spend 35 hours per week sat close to someone, you'd better have things to share with him.

Interaction also means you are invited to knock at some random office and discuss with someone. By sharing ideas and confront them, it usually happens that new solutions arise. Moreover it is always a good point that knowledge is spread. But pay attention while doing this to not interrupt someone in the middle of his work. The rules are:

- door open: feel free to interrupt me
- door closed: leave me alone (at least check if I am not in the middle of a pomodoro session)

### 1.2 Stay in Touch When Working Remotely

Working at home or outside of the building is fine as long as you are still present in the team. Actually even more than when you are physically present.

Being present while working remotely means answering mails regularly, being connected to the team IRC channel and Google Talk. In addition, it can not hurt to be connected to Jabber, Skype, or any protocol where you have a colleague as contact.

### 1.3 Pair Program

Pair programming is a fun way to work. And without noticing great things could be achieved more easily when you are not alone in front of your screen. It is also a good way of tackling difficult and/or boring tasks.

It is also a tool used to teach someone about a topic he may not be aware of without the need of a long introduction. When you feel in pair-programming mood, announce it on the team mailing-list. You can also plan it in advance to be sure to have a wingman.

In addition, friday afternoon is dedicated to pair programming. This *mandatory* event is meant to share knowledge about internal projects and to fix Pharo bugs.

### 1.4 Attend to Meetings

Even if meetings may sound boring, it is usually a good way to share knowledge and experience. You can easily collect new ideas by attending to meetings and interact with the speaker.

On the other hand, one should not forget that some meetings are mandatory (including the "Monthly Pharo Meeting"). Part of your job is to be aware of the direction Pharo is taking and to collaborate to the planning of Pharo vision.

## 2 Contribute Respecting the Community

Pharo is a collective development. It means that each team member can contribute to any part of Pharo. To do so, the only way is that all the projects have to be managed the same way. Thanks to this, the effort needed to be introduced to a new project is minimal. This management takes several ways.

### 2.1 Read the Mailing List

In addition to the internal mailing list, the Pharo community lives through the pharo-dev and pharo-users mailing lists. You are expected to follow at least the pharo-dev list and to advocate the Pharo vision. Since this is the main public Pharo window, you have to watch your words and to keep your personal beliefs about the Pharo vision for the internal mailing list.

Do not forget we are the Pharo ambassadors.

## 2.2 Use the Issue Tracker

Pharo is using fogbugz[1] to keep track of issues for this, our brains are too small to remember such details. One of the particularities of this issue tracker is to provide project support. The idea is to have a fogbugz project for each Pharo project in the issue tracker.

You are responsible of the issues related to your projects, but you are also responsible of the overall Pharo situation. You are invited to review cases and provide fixes when you can.

The way to report a Pharo issue is to create a new case on fogbugz, see the fogbugz documentation for more details. Issues only reported by mails or orally will be denied. The format to propose patches is a slice or a working configuration. Any other format will be denied by default. Only if the first two approaches fail, we fallback to manually create and load changesets.

## 2.3 Commit Often

Committing often is the only to assure multiple people are not doing the same thing at the same time. Or worse, that multiple people are producing conflicting code.

Do not be afraid of committing broken code. Committing should be considered as saving in the cloud. The tip about committing is to commit each time you finish to implement something (even a really tiny new behaviour), each time you save your image and each time you leave your machine (for a break, a coffee, or when you leave the lab). Providing working code is the concern of configurations, see section 2.6.

It also contributes to make more people aware of the project and the way it is going. This leads to an overall better quality of the project (the more reviews the better).

It brings visibility to your work and to Pharo, showing the community is active.

## 2.4 Run Tests

At one point of a project life, it will have to be integrated into Pharo itself. To do so we first have to be sure it will fit the current state of the system. The tool used to ensure this is unit tests. Tests show the health of a project at different levels.

They first bring validation about the project behavior itself. With tests, you can easily detect the introduction of bogus code.

Secondly, they ensure the integration of your project inside Pharo. Even if your project is working, some dependencies or side effects could affect your project's validity.

[1] `http://pharo.fogbugz.com`

Finally, they are sources of examples of the typical use cases of your project. By looking at tests, one can learn how the different objects from your project are interacting with each other.

## 2.5 Write Documentation

Documentation is critical for the understanding of your project, how it is composed and how it should be used. Most of the time having proper methods names, protocol classification and class comments is enough. For Pharo projects these requirements are mandatory.

Good method names are crucial since a method is the smallest granularity of source code. They should be self-explanatory. If you can not achieve this because it would lead to a too long name, then you can tackle this by writing a method comment explaining the purpose of it. It is always better to have a longer method name than having to look at its source code to know what it does.

Protocols are used to classify methods by groups of meaning or usage. Having good names and good classification reduces the time spent to find the needed methods and help to understand its purpose.

Class comments are often the first piece of documentation a user will read. Be sure to put the maximum number of relevant information there. A good class comment also includes an example which shows the purpose of the class and the design pattern applied.

It is important to write the documentation along the implementation for two reasons. First it helps to have a better mental representation of the interactions between objects. Second it helps contributors to understand what you did in your last commits.

If your project is not documented, it will NOT be integrated into Pharo.

## 2.6 Use Configurations

A configuration is a map representing the way to load a project and its dependencies. By providing a configuration, you ensure anyone can load your project, and contribute to it.

A configuration also helps by supporting symbolic versions which mean you can specify which version of your project is the stable one (the one to be used for regular users) and the development version (the version used by developers, often the bleeding edge).

Configurations are also used for continuous integration builds.

Your *stable* version should always be a working version. If your project does not have a configuration, it will NOT be integrated into Pharo.

### 2.7 Build Jenkins Jobs

Pharo development is based on continuous integration provided by Jenkins[2]. Jenkins also manages all the team projects. It means each team project must have a corresponding Jenkins job.

This is needed to ensure that Pharo improvements are not breaking your code (and vice versa). It helps tracking introduction of bugs by giving fast feedback.

If your project does not have a Jenkins job, it will NOT be integrated into Pharo.

### 2.8 Backup your Work

There is nothing more annoying and discouraging that losing your work. In order to reduce the risk of such accidents, each team member should backup its all machine at least once per day. Backup hard drive are provided so you just have to think about plugin it while arriving.

Backuping your hard drive provide the insurance your data are saved. It also allows one to retrieve an older version of a file, or a file deleted by accident. Moreover laptop may have product failures and the hard drive can lose its contents. Without a decent backup, everything will be gone forever.

## 3 Your rights

This section is a short reminders of your rights as a team member.

### 3.1 Be focused

When get close from a deadline, one may want to be focused on its own tasks. If you announce your deadline and your will to focus on it in the mailing list, then no one have the right to interrupt you anymore.

Also if you are doing a pomodoro session or a meeting and you do not want to be interrupted, you have the right to stick a sheet of paper saying it on your closed door. This way people will now what you are doing, and that you need to keep focus.

### 3.2 Be tired

It happens to everyone to be grump, tired, low motivated, or frustrated. Do not hesitate to tell it, to discuss it. Solutions can only be found through constructive dialogs (and we may have some hidden candies for such occasions).

Being a team also means taking care of each other.

[2] `https://ci.inria.fr/pharo-contribution/`

# 4 Bibliography

A team culture is built on shared experiences and also on the fact that people are using the same language. Some books are considered as "must have been read", so anybody can assume that everyone else have the same references when talking about a topic.

Those books are available for every one to be read and read again.

We present here a list of books very team member should have read:

- Pharo by Example, Black, Ducasse, Nierstasz, Pollet
- Smalltalk Best Practice Patterns, Beck
- The Design Patterns Smalltalk Companion, Alpert, Brown, Woolf