# Project Management Beyond Agile

Series of lectures by Yegor Bugayenko to be presented to students of Innopolis University in 2023. and video recorded

The entire set of slide decks is in yegor256/pmba GitHub repository.

Abstract:
Today, Agile has emerged as a widely-used term among managers overseeing software development projects. Nonetheless, it's important to note that Agile is not a management framework per se, but rather a set of guiding principles intended for managers already utilizing an established framework, such as IBM's RUP® or Microsoft's MSF®. Furthermore, the PMBOK™ by PMI® posits that project management is a deterministic endeavor, regulated by stringent rules and even laws. This course seeks to form a connection between the traditionally dry formalism of project management and the progressive practices of Agile/XP.

**What is the goal?**

The main aim of this course is to enable students to comprehend the core principles of project management as outlined by PMBOK™. It further encourages them to implement these principles in practical scenarios, particularly in commercial and open-source software development projects.

**Who is the teacher?**

Yegor is developing software for more than 30 years, being a hands-on programmer (see his GitHub account: @yegor256) and a manager of other programmers. At the moment, he is a director of an R&D laboratory in Huawei. His recent conference talks are in his YouTube channel. He also published a few books and wrote a blog about software engineering and OOP. He previously taught a few courses in Innopolis University (Kazan, Russia) and HSE University (Moscow, Russia), for example, SSD16 (2021), EQSP (2022), PPA (2023), COOP (2023) (all videos are available).

**Why this course?**

Agile, viewed as a software development philosophy, can be highly effective when applied by those well-versed in essential project management principles, such as scope, cost, and risk management. However, as Agile's popularity rises, there's an observed decline in the understanding of project management as a scientific discipline, a trend noted among both new graduates and experienced software engineers and managers. This course aims to enhance such understanding, minimizing the tedium typically associated with traditional management disciplines.

**What's the methodology?**

Each lecture engages in an analysis of several practical scenarios within a software development team. The aim is to discern both productive and unproductive situations. From these observations, conclusions are drawn that help students gain a clearer understanding and improved perspective of their own management decisions.

## Course Structure

Prerequisites to the course (it is expected that a student knows this):

- How to write code
- How to design software

After the course a student *hopefully* will understand:

- How to draw a Gantt Chart and what for?
- How to fire an under-performing team member?
- How to create and maintain a Risk List?
- How to identify risks in a project?
- How to do quantitative and qualitative risk analysis?
- How to report project status to a project sponsor?
- How to estimate project costs?
- How to calculate project budget?
- How to avoid “Gold Platting”?
- How to decompose project scope into work packages?
- How to measure performance of each team member?
- How to optimize critical path using CPM?
- How to work with a traceability matrix?
- How to specify requirements unambiguously?
- How to organize the work of a Change Control Board?
- How to motivate programmers for higher productivity?
- How to structure a software development contract?

## Lectures & Labs

The following topics are discussed:

1. Integration Management
    - How to read PMBOK?
    - How to identify and specify the problem?
    - How to establish project rules?
    - How to organize decision making process?
    - How to embrace the chaos?
    - How to be a good manager?
2. Scope Management
    - How to set Definition of Done (DoD)?
    - How to decompose a project to tasks?
    - How to avoid Gold Platting?
    - How to blame the product not yourself?
    - How to avoid micro-management?
3. Time Management
    - How to avoid Daily Stand-ups?
    - How to stay away from Gantt Charts?
    - How to avoid playing Planning Poker?
4. Cost Management
    - How to estimate project budget?
    - How to pay what they deserve?
    - How to pay 10x to 10x programmers?
    - How to stop paying for your team education?
5. Quality Management
    - How to differentiate QA from testing?
    - How to organize code reviews?
    - How to get rid of altruism?
    - How to aim for speed instead of quality?
6. *Human* Resource Management
    - How to not spend two hours for an interview?
    - How to motivate people?
    - How to measure productivity of a dev team?
    - How to measure productivity of a research team?
    - How to boost team morale?
    - How to fire painfully?
7. Communications management
    - How to avoid technical meetings?
    - How to use Ticket Tracking Systems?
    - How to avoid emails?
    - How to work remotely?

- How to enjoy turnover of talents?
- How to punish your team?

8. Risk management
    - How to build a risk list?
    - How to predict and prevent failures?
    - How to respect and not trust your team?
9. Procurement Management
    - How to supervise an external team?
    - How to avoid hourly pay?
    - How to control their quality?
    - How to measure productivity of an external team?
10. Stakeholder Management
    - How to be a good office slave (for your boss)?
    - How to make your boss happy?
    - How to be honest with a customer?
    - How to bill incrementally?
    - How to pass PMI exam?

At the end of each lecture students will be asked to create a project management *artifact* for their own project. The artifacts may also be completed at the Labs and then presented for review to the Teaching Assistant.

## Grading

Students have to form groups of 2–4 people in each one.

Each group should pick a research topic from the list suggested by the teacher at the first lecture. A group is allowed to pick its own topic, if approved by the teacher at the first lecture.

Each group should do a research and write a research paper, according to this recommendation. The length of the pager may be no longer than four pages acmart/sigplan format (two columns).

A group may ask another group to peer-review their paper (no more than twice).

Before the end of the course, the paper must be submitted to ICSE, ESEC/FSE, or a similar A* conference (student tracks and workshops are allowed). The paper may be submitted only after the teacher approves it.

There is no exam at the end of the course. Instead, each student earns points for the following:

| | |
|---|---|
| Attended 75%+ of all lectures | +1 |
| Completed 50%+ of all artifacts | +1 |
| Completed 75%+ of all artifacts | +1 |
| The paper positively peer reviewed | +1 |
| The paper submitted to a conference | +1 |
| The paper positively reviewed by the teacher | +4 |

Then, 8+ points means “A,” 6+ means “B,” and 4+ means “C.”

## Learning Material

The following books are highly recommended to read (in no particular order):

Rita Mulcahy, *PMP Exam Prep, 9th Edition*, 2018

Rita Mulcahy, *Risk Management*, 2010

Karl Wiegers et al., *Software Requirements*

Alistair Cockburn, *Writing Effective Use Cases*

Steve McConnell, *Software Estimation: Demystifying the Black Art*

Frederick Brooks Jr., *Mythical Man-Month, The: Essays on Software Engineering*

David Thomas et al., *The Pragmatic Programmer: Your Journey To Mastery*

Robert C. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*

Jez Humble et al., *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*

Michael T. Nygard, *Release It!: Design and Deploy Production-Ready Software*

Yegor Bugayenko, *Code Ahead*, 2019

Blog posts of Yegor Bugayenko, on his blog