Project Management Beyond Agile

Series of lectures by Yegor Bugayenko. The entire set of slide decks is in yegor256/pmba GitHub repository. Previous lectures were video recorded.

Abstract:
Today, Agile has emerged as a widely used term among managers overseeing software development projects. However, Agile is not a management framework per se. Instead, it is a set of guiding principles intended for managers already utilizing an established framework, such as IBM's RUP® or Microsoft's MSF®. The PMBOK™ by PMI® posits that project management is a deterministic endeavor, regulated by stringent rules and even laws. This course seeks to bridge 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™. The course encourages them to implement these principles in practical scenarios, particularly in commercial and open-source software development projects.

Who is the teacher?

Yegor has been developing software for more than 30 years. He is a hands-on programmer (see his GitHub account: @yegor256) and a manager of other programmers. At the moment, he is a software developer in Huawei. His recent conference talks are in his YouTube channel. He also published several books and wrote a blog about software engineering and OOP. He previously taught several courses at Innopolis University (Kazan, Russia) and HSE University (Moscow, Russia). Examples include SQM, EQSP, PPA, and COOP (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. This trend has been noted among both new graduates and experienced software engineers and managers. This course aims to enhance such understanding while minimizing the tedium typically associated with traditional management disciplines.

What's the methodology?

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

Course Structure

Course prerequisites (students are expected to know this):

- How to write code
- How to design software

After the course, students will *hopefully* understand:

- How to draw a Gantt Chart and for what purpose?
- How to fire an underperforming 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 Plating”?
- 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 the 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 into tasks?
    - How to avoid Gold Plating?
    - 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 avoid spending two hours on 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. Communication 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 the PMI exam?

Grading

To pass the course, students must attend lectures and earn points by contributing to any of these open-source projects:

- `objectionary/lints` (Java + XSLT)
- `cqfn/aibolit` (Python)
- `objectionary/eoc` (JavaScript)
- `yegor256/0rsk` (Ruby)
- `objectionary/reo` (Rust)

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

Result	Points	Limit
Attended a lecture	+1	6
Submitted a ticket (that was accepted)	+2	8
Submitted a pull request (that was merged)	+4	32

Final grades are based on accumulated points:

- 25+ earns an “A,”
- 17+ a “B,” and
- 9+ a “C.”

If eight or fewer points are accumulated during the course, the student may still earn them before the retake.

An online lecture is counted as “attended” only if a student was personally present in Zoom for more than 75% of the lecture’s time. Watching the lecture from a friend’s computer *doesn’t* count.

Learning Materials

[1] Frederick P. Brooks. *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley, 1978. DOI: 10.5555/540031.

[2] Yegor Bugayenko. *Code Ahead*. Amazon, 2018.

[3] Alistair Cockburn et al. *Writing Effective Use Cases*. Addison-Wesley, 2000. DOI: 10.5555/517669.

[4] Jez Humble et al. *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*. Pearson Education, 2010. DOI: 10.5555/1869904.

[5] Andrew Hunt et al. *The Pragmatic Programmer: From Journeyman to Master*. Pearson Education, 1999. DOI: 10.5555/320326.

[6] Robert C. Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall, 2008. DOI: 10.5555/1388398.

[7] Steve McConnell. *Software Estimation: Demystifying the Black Art*. Microsoft Press, 2006. DOI: 10.5555/1204642.

[8] Rita Mulcahy. *PMP Exam Prep*. RMC Publications, 2009.

[9] Michael Nygard. *Release It!: Design and Deploy Production-Ready Software*. Pragmatic Bookshelf, 2007. DOI: 10.5555/1200767.

[10] David West. *Object Thinking*. Pearson Education, 2004. DOI: 10.5555/984130.

[11] Karl E. Wiegers et al. *Software Requirements*. 2nd ed. Microsoft Press, 2003.

Blog posts by Yegor Bugayenko on his blog may also help.