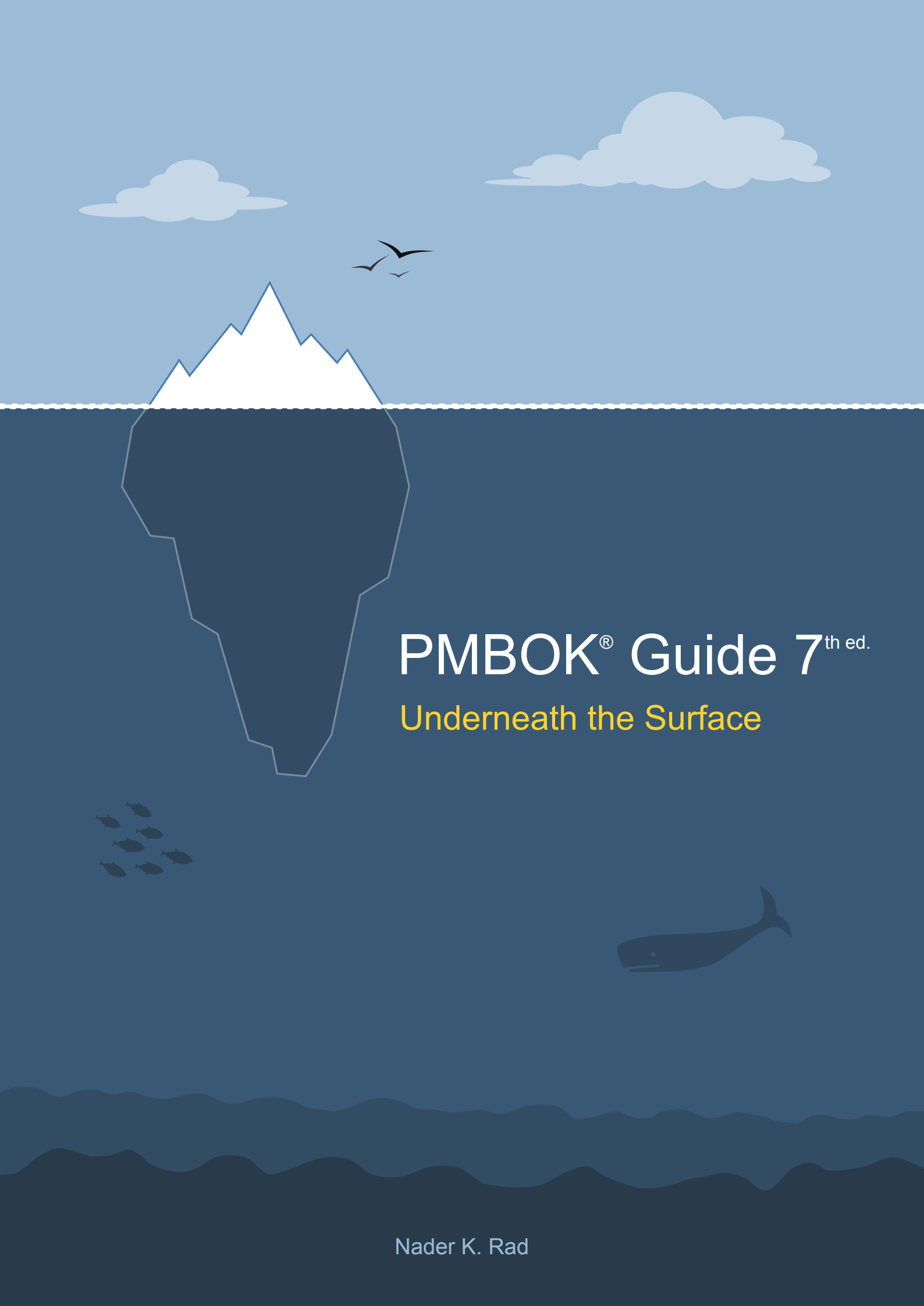

PMBOK® Guide 7th ed.

Underneath the Surface

Nader K. Rad

This file has been created automatically based on the latest version of the book on 2023-06-08. For the latest version and other formats, visit the website: https://pmbok.guide/

About the Author

I started working with projects about 25 years ago: first, as a project planner in construction projects, and then I moved to large process-plant projects. In parallel, I was also involved in IT projects and startups.

I gradually moved from project planning into the wider project management domain by helping project managers and companies improve their project, program, and portfolio management systems.

Nowadays, I'm spending most of my time developing eLearning courses in our small company, Management Plaza (https://mplaza.training), and contributing to standards:

- **PMBOK® Guide 7th edition** - Core development team member
- **P3.express** - Core development team member
- **NUPP** - Core development team member
- **PRINCE2® 6th edition** - Reviewer
- **PRINCE2 Agile®** - Reviewer
- **MSP®** - Reviewer
- Etc.

You can find out more about me here:
https://nader.pm or gemini://nader.pm

Disclaimer

Everything you read in this book is my personal opinion rather than that of PMI or the core development team of the PMBOK Guide. In fact, I'm sure at least some of them would disagree with some of the content in this book!

After all, if no one disagrees with you, you've probably not said anything useful.

IP Statements

This book mentions multiple resources besides the PMBOK Guide. Of these, PM², P3.express, micro.P3.express, Scrum, and NUPP are open, non-proprietary resources, while the others are proprietary ones protected by copyright. The following are their IP statements:

1. What is the PMBOK Guide?

In the 1960s, project management became more serious and recognized as a specialty in the US, and people started organizing conferences and events around it. In 1969, a few of those people formed a not-for-profit organization named ***Project Management Institute (PMI)***.

1.1. The certification program

Besides organizing events and bringing project managers together, in 1984, PMI also created a certification program called ***Project Management Professional (PMP)***, which is still one of the best-known and most desirable certifications in this domain.

1.2. The structured guide

The success of the PMP exam, which was formed around general project management knowledge, encouraged PMI to document and structure that knowledge. The result was ***A Guide to the Project Management Body of Knowledge***, or the ***PMBOK Guide*** for short. It became one of the main resources for the PMP exam, although not the only one.

Following on from a white paper in 1987, the first edition was published in 1996, and since then, there has been a new edition every four or five years:

- **1996:** PMBOK Guide
- **2000:** PMBOK Guide, 2000 edition
- **2004:** PMBOK Guide, 3rd edition
- **2008:** PMBOK Guide, 4th edition
- **2013:** PMBOK Guide, 5th edition
- **2017:** PMBOK Guide, 6th edition
- **2021:** PMBOK Guide, 7th edition

The first edition described project management with a set of 37 processes, each

with various ***inputs***, a few ***tools and techniques***, and certain ***outputs***, along with some extra information about the context within which projects are run. Those processes were categorized in two different ways:

- **Process Groups:** These categorize the processes based on their role in the overall/typical project management life cycle.
  - Initiating
  - Planning
  - Executing
  - Controlling
  - Closing
- **Knowledge Areas:** These categorize the processes based on their subject.
  - Integration
  - Scope
  - Time
  - Cost
  - Quality
  - Human resource
  - Communications
  - Risk
  - Procurement

The next five editions followed the same structure and refined the content. The number of processes grew from 37 to 49, which was mainly a result of breaking down the existing ones into more detailed processes. The ***communications management*** knowledge area was later divided into ***communications management*** and ***stakeholder management***. Some of the elements were renamed to be clearer, more precise, more general, etc.; for example, ***human resource management*** became ***resource management***. Finally, the whole guide grew from 176 pages in the first edition to 756 pages in the sixth edition.

The seventh edition, on the other hand, was written from scratch, and with a ***principle-based*** approach instead of the ***process-based*** one described above. It's more inclusive of different projects and is only 274 pages long.

Throughout the book, you will learn more about the reasons for this fundamental change. However, this book is designed to explain what the 7th edition of the PMBOK Guide is and doesn't aim to compare it with the previous editions.

The old processes of the PMBOK Guide don't exist in the latest edition anymore. However, some people are very attached to them, and for this reason, they are now contained in a separate publication called ***Process Groups: A Practice Guide***.

1.3. The extension of the family

Based on the success of the PMBOK Guide and the PMP exam, PMI continued expanding in the following ways:

- **Certification programs**
  - Certified Associate in Project Management (CAPM)
  - Disciplined Agile Coach (DAC)
  - Disciplined Agile Scrum Master (DASM)
  - Disciplined Agile Senior Scrum Master (DASSM)
  - Disciplined Agile Value Stream Consultant (DAVSC)
  - PMI Agile Certified Practitioner (PMI-ACP)
  - PMI Professional in Business Analysis (PMI-PBA)
  - PMI Project Management Ready
  - PMI Risk Management Professional (PMI-RMP)
  - PMI Scheduling Professional (PMI-SP)
  - Portfolio Management Professional (PfMP)
  - Program Management Professional (PgMP)
- **Fundamental standards**
  - The PMI Guide to Business Analysis
  - The Standard for Earned Value Management
  - The Standard for Organizational Project Management
  - The Standard for Portfolio Management
  - The Standard for Program Management
  - The Standard for Risk Management in Portfolios, Programs, and Projects
- **Practice standards**
  - Practice Standard for Project Configuration Management
  - Practice Standard for Project Estimating
  - Practice Standard for Scheduling
  - Practice Standard for Work Breakdown Structures

    - Project Manager Competency Development Framework
- **Practice guides**
    - Agile Practice Guide
    - Benefits Realization Management: A Practice Guide
    - Business Analysis for Practitioners: A Practice Guide
    - Governance of Portfolios, Programs, and Projects: A Practice Guide
    - Managing Change in Organizations: A Practice Guide
    - Navigating Complexity: A Practice Guide
    - Process Groups: A Practice Guide
    - Requirements Management: A Practice Guide

Besides the ***PMI-ACP*** certification program, none of these attained any level of popularity comparable to the PMP exam and the PMBOK Guide.

1.4. The nature of the PMBOK Guide

We can categorize any resource about project management in one of the following groups:

- **Methodologies,** which give you a path through the project.
- **Guides,** which help you to be more efficient in using the methodologies.

What you need first and foremost is a methodology; and then ***after*** implementing a methodology, you can use the guides to enrich it.

The PMBOK Guide is designed to be a ***guide*** rather than a ***methodology***. However, the ***process-based*** nature of the older versions was confusing for many and led them to think of the guide as a methodology, but when they tried to use it as such, it never gave good results. That's why the old versions had a disclaimer in their introductions explaining that they're not methodologies and that people need to select and use a methodology in parallel with it. One of our goals in the 7th edition was to prevent this from happening, and I think we succeeded, in that the new, ***principle-based*** approach of PMBOK 7 cannot be mistaken for a methodology.

The PMBOK Guide and its complementary resources help you to develop a better methodology by

- helping you interpret the system using the **principles**, and
- helping you enrich the system using the **performance domains**.

It's for these reasons that the book contains the following main chapters:

- **Understanding and Interpreting** goes over how the PMBOK principles help you gain a better understanding of project management, which can be used when selecting, implementing, and using a methodology.
- **Selecting a Methodology** introduces a few methodologies to give you an idea of what it is that you need to be more effective in.
- **Enriching the Methodology** explains how the PMBOK Guide can help you enrich the methodology you've implemented.

1.5. The process of developing the PMBOK Guide

The PMBOK Guide and other standards in PMI follow a development process that complies with good practices for standardization according to the American National Standards Institute (ANSI) and to some degree to the International Organization for Standardization (ISO).

The core development team of PMBOK 7 consisted of 12 members (including this author) with different backgrounds and specialties, all of whom were responsible for the content. There was a leadership team, responsible for coordinating, reflecting the expectations of PMI in the development team, etc.

There were a select group of about 70 reviewers who helped by reviewing and commenting on major deliverables throughout the work. At the end, an exposure draft was made available to the public to review. More than 600 people helped by reviewing the draft and submitting comments. The development team used those comments to improve the content, and the final version was published after this.

2. Understanding and Interpreting

A few years ago, I was reading a collection of essays by the famous physicist, Richard Feynman, in which he explained a problem he called ***Cargo Cult Science***. I couldn't stop wondering how well that explains a fundamental problem we have in project management when I was reading it. So, let me explain:

There were a few isolated tribes of people in New Guinea and a few other islands. During the two World Wars, they saw cargo planes landing and bringing goods for the temporary military bases on their islands. Seeing those large human-made flying objects was magical for them. Moreover, the soldiers used to give away or trade some of the cargo, which was fascinating to the islanders.

When the wars ended, there were no cargo planes anymore, and the islanders wanted them back. So, they built runways, control towers, and mock planes. They wore wooden headphones with antennas, marched around the mock airbase, and lit the sides of the runways, thinking that these should attract the flying cargo planes and convince them to land.

In their mind, simulating what others were doing was enough to have the same result. In reality though, as you can imagine, it wasn't enough to bring back the cargo planes.

Doesn't that sound familiar?

Many people look at successful projects and then start imitating them. They think as long as they have something that looks like a plan, a business case document, or a project charter document, etc., they have proper project management and they can get the results they need. This is specially the case in many Agile projects, where they think as long as they use the Scrum labels and have big boards with lots of sticky notes on them, they are good Agilists.

What's missing in many projects is the essence of what we do and the real dynamics behind the scenes. The rituals, documents and everything else are not the end, but the means to the end.

Having principles is a way to bring the attention back to what we really need to

have. We think about the principles, and then carry on producing various project management outputs that serve different purposes without blindly imitating others. That's why the 7th edition of the PMBOK guide has become ***principle-based***.

Principles are the main content of PMBOK 7. It's fair to say that PMBOK 7 is mainly about how you can avoid the Cargo Cult effect! We'll explore the PMBOK principles in detail in the rest of this chapter, and then we'll also have a short overview of principles from other resources to give a wider perspective.

2.1. PMBOK 7 principles

There are 12 principles in PMBOK 7:

1. Be a diligent, respectful, and caring steward.
2. Create a collaborative project team environment.
3. Effectively engage with stakeholders.
4. Focus on value.
5. Recognize, evaluate, and respond to system interactions.
6. Demonstrate leadership behaviors.
7. Tailor based on context.
8. Build quality into processes and deliverables.
9. Optimize risk responses.
10. Navigate complexity.
11. Embrace adaptability and resiliency.
12. Enable change to achieve the envisioned future state.

An element that can be considered missing in this list is ***ethics***. There's no item for ethics in this list because ethics is at a higher level of abstraction and applies to everything else. Regardless of how we formulate it, it's essential to be mindful of ethics, and a good resource is the ***PMI's Code of Ethics & Professional Conduct***: https://www.pmi.org/codeofethics

2.1.1. Be a diligent, respectful, and caring steward

If you ask hundreds of project team members what they think about the project managers they've met in their career, at least some of them would tell you horror

stories about annoying project managers who didn't understand anything about the work and only bossed people around and had unrealistic expectations of them. For some reason, you can hear a lot of these sad stories in IT development projects, which is a reason why some of the newer methods in IT development are against having a central project manager role.

On the other hand, we've all seen many organizations where they appoint the most senior expert as the project manager because of their deep expertise in technical aspects. Many even think that a good technical expert must be promoted to a (project) manager sooner or later. In these cases, the organization loses a great technical expert and gains a mediocre manager.

2.1.1.1. Promotion vs. career change

To make things clear, we have to start with the management concept. When a technical expert becomes a manager, it's not a ***promotion***, but a change in ***career***. Being good at technical aspects doesn't guarantee that you can be a good manager because you need a different set of skills and attitudes for management. In fact, many people aren't really comfortable having management roles and prefer to remain technical – we've been ruining their professional lives by forcing them into those positions.

A better way is to let technical people remain technical and seek other ways of promoting them, while in parallel investing in recruiting and nurturing other people for management positions. The latter can be enriched by undergoing higher education in (project) management as well.

2.1.1.2. A true project manager

We also need to rethink the role of project manager. A proper project manager is not a "boss" (there are already enough bosses in every organization), but rather a supporter who helps create a safe and productive environment for the team members where they can carry out their expert activities better. Project managers must be focused on coordination, problem solving, conflict resolution, etc.

My usual advice is that if you feel that the label "project manager" has a negative connotation in your organization because of past experiences, lose the label and

simply call it ***chief project facilitator***. It helps people who take that role by continuously reminding them of their main responsibility as well.

2.1.1.3. Understanding the technical aspects

So, when you think of the project manager role as a non-technical supporter rather than a technical boss, it's clear that we don't expect a project manager to plan and control the project alone, because they can't know enough about the project to do so. Therefore, instead of doing those things by themselves, we expect them to facilitate team members to create well-formed plans and come up with proper responses to deviations when needed. That's proper project management.

Should project managers have technical knowledge?

We can think of three responses to this:

1. Yes, project managers should have technical knowledge.
2. Project managers don't have to have technical knowledge, but it may help if they do.
3. No, it's not necessary, and in fact it's even better if they don't have technical knowledge.

The first response is an old-fashioned way of thinking that causes many problems. The second one is a relatively modern, yet slightly conservative way of thinking, compatible with what I explained before, and I believe we can say that PMI believes in it as well. The last one is a progressive idea that many people would find difficult to accept, but that I'm in favor of.

Imagine you're managing a project and you don't know anything about its technical aspects. The only thing you can do is to help the technical people do their best by focusing on facilitation, coordination, etc. If there's a complicated technical decision and you're not sure about the opinion of the team members, you can always consult one or a few external experts, and it should be fine.

However, imaging managing a project in an area in which you're also a technical expert. If you're like most human beings, you won't be able to stay quiet when you see the team members doing something imperfectly, and you'll get yourself

involved in the technical aspects. In doing so, you're not just another technical expert chipping in, but your position as a project manager may also give you an advantage, which can gradually turn you into one of those annoying project managers who micromanage everyone. That's why the third option of choosing a project manager who doesn't have technical knowledge can be preferable.

2.1.1.4. Acceptance of non-technical project managers

If you accept the previous argument, you may still have two big concerns. The first one is that while this may be a better setup, people still find it difficult to accept project managers who are not technical experts. If you're such a project manager, note that the objection is because they understand the project manager as someone who's supposed to be the main technical decision maker.

So, you should explain at the beginning that you're not going to be their decision maker, and that they are the technical experts responsible for those decisions. You also need to explain to them that you're there to make working easier, more pleasant, and more productive for them. Of course, talking is easy, and you have to prove it to them by your actions. If you do it right, after a few weeks or months, they will trust you.

A company I knew once assigned a very experience project manager to an IT development project. The project team objected to him because he had only worked in construction projects. I explained everything I've said above to them, and asked them to give him a try. I returned there a few weeks later and asked how it was with the project manager. They were all happy about it, and one person told me, "I had no idea working in a project could be so easy and comfortable! I don't know how I'm going to go back to working on typical projects after this."

2.1.1.5. What to do when you're a technical expert

The second concern is that you may be appointed as the project manager, and you do have technical knowledge; so what should you do? The only thing you can do is to be aware and conscious of it, and draw a line between the technical and managerial aspects and don't get yourself involved in the technical aspects at all.

One of the great project managers I know was an architect working on construction projects. In one of the meetings, two companies involved in the project had a disagreement about the architectural design of an area and were arguing about it. Suddenly, one of them asked the project manager what he thought about it, because after all, he was also an architect. He responded, “I’m not here as an architect, but as a project manager. As a project manager, I can tell you that it’s an important decision, and I’d like to ask you to have a workshop in the following days, explore and document all aspects, make a decision, and send me a report about it. It’s best to finalize the decision by the end of the week, because otherwise we’ll have a delay in our project. Shall we move on to the next topic?”

If you accept the project management role, you need to have the mental discipline to do as he did!

2.1.1.6. What remains for the project manager?

So, when we leave the technical aspects to the technical roles, what remains for the project manager other than the relatively obvious things such as facilitation and coordination?

Well, for example, you should feel responsible for creating an environment where team members can grow in their careers. Yes, that’s your ***responsibility***.

There are many other things you should do. For more ideas, I can recommend the following:

- PMI’s Code of Ethics & Professional Conduct: https://www.pmi.org/codeofethics
- P3.express Practitioners Code of Conduct: https://p3.express/certification/code-of-conduct/
- IPMA’s Individual Competence Baseline (ICB): https://www.ipma.world/individuals/standard/

In short, you should care for the people in the project as well as the resources entrusted in the project, have integrity, be trustworthy, and comply with the rules, regulations, and requirements (as long as they are compatible with the ethical rules).

Lastly, to be able to do all of these, there's one tool you'll need most: being a good listener. Don't listen to respond, but listen to understand. Don't wait for people to come and talk to you, but initiate the conversation and actively seek problems to solve or areas to improve – it's always easier to fix things the sooner you find them.

2.1.2. Create a collaborative project team environment

What we need for success is not a number of great people in the project, but a great project team.

So, how would you do that?

2.1.2.1. Secret team building

Imagine that you need to plan something. To do that, you can go to five team members, collect the data, and use that to create the plan. An alternative is to conduct a facilitated workshop with people who can help, and do it together. The second option, when done right, is an opportunity to teach them how to work together, improve their relationships, and move towards a stronger team.

Forget about the boring, corporate-style team-building activities; any time you want to do something, you should ask yourself whether it's possible to adjust it in such a way that it could work as a team-building activity as well. In many cases, you can make that happen.

2.1.2.2. Satisfaction evaluations

As another example, the P3.express methodology has monthly cycles, with a number of closing activities at the end of each month. One of them is to evaluate stakeholder satisfaction, where you evaluate both the satisfaction of the internal team members and the external stakeholders, and the other is for using that information and designing improvement plans. The recommendation for the second activity is to bring all the team members together, give them the information, and use a proper facilitation technique such as ***Delphi*** to let them design those improvement plans.

Such a process has multiple advantages, including the following:

- You will get great ideas you could never have come up with on your own or in a small group.
- People will feel appreciated, and because they are the ones designing the improvement, you will have their buy-in and a higher chance of success for the plan.
- It will improve their collaboration in their other tasks.

2.1.2.3. Common understanding

Depending on the project environment, you may need to do other things as well to encourage collaboration. For example, when the project is not too small, it would work better to make it clear what is expected from each person, and what they can expect from others. What I just mentioned is one of the principles in PRINCE2® (a project management methodology). Going further, you may even need to formalize the flow of works in the project; e.g., each design document has to go to ##### for peer review; the peer reviewer should respond in less than ##### days; then it goes to…

Besides responsibilities and flow of work, you may need to make the basic behaviors clear too, with some ***ground rules*** that everyone will respect when working together.

You may be thinking it may be bossy to tell people all of these things, and as such, a violation of the previous principle. That would be true if ***you*** set all of those things. The best way, however, is to facilitate the team to set those rules themselves whenever possible. That way, the rules will be more realistic and the acceptance of the team members higher. For example, the P3.express Code of Conduct that was mentioned above was developed and adjusted by the community of practitioners rather than being dictated from the top downwards.

2.1.3. Effectively engage with stakeholders

Some people use the word “stakeholder” to refer to just the customer. In the project management literature, we define it as any person or group of people who have an interest in the project and can impact it. In that sense, the team

members, client, suppliers, competitors, regulators, and many other people and groups are stakeholders.

2.1.3.1. Forgotten stakeholders

Once, I worked for a wealthy organization that, besides running its own projects, used to fund a few not-for-profit projects as well. One of them I was aware of was a project run by a small organization to build a shelter on top of a mountain. It was specially interesting to me because I loved hiking back then. (I really miss hiking now that I live in one of the flattest places on Earth!)

The representatives of that small organization used to come to our office once a month, submit a report of what they'd done so far and another of what they were going to do, with an estimate of its cost, and we used to give them the money. One day, I realized I'd not seen them for a while. I asked around and realized that they hadn't submitted any new reports for a while. When we called them, they told us that the national cultural heritage organization had stopped their project because they were building it on an ancient road!

The ancient road was hardly visible, and you could easily have mistaken it for a track organically created by hikers. A lot of money was wasted on that project (building something on top of a mountain is ***very*** expensive), and a historic artifact was damaged as well. So what went wrong?

Before starting the project, they could have gone to the spot and asked the hikers what they thought about building a shelter there. Some of the experienced hikers would have known about the road and could have told them. They could even have got other useful information by doing this. Better still, they could have put a notice up there, announcing that their organization was going to build a shelter there and asking the hikers to send their comments.

Having this type of concern is what separates a great project manager from a good or an average one. Some people would object that you're wasting time doing this kind of thing instead of starting work immediately, but you'd know better that it can help prevent a lot of troubles in the long run. It's not about how soon you start the work, but how soon you finish it, and with what quality, cost, etc.

2.1.3.2. Communications

An essential concern in project management is to identify stakeholders and engage them in the project. Identifying stakeholders is not easy, and you have to put effort into it. Don't forget to get help from the team members when identifying stakeholders, and do it in a way that works as a team-building activity as well (as also mentioned in the previous principle).

As for the stakeholders you've identified, you need to frequently communicate with them and actively ask for their input. The format of communication depends on the type of stakeholder, and you should have various ways tailored to each group. An extreme example I can think of is from many years ago, when someone started constructing a low-cost apartment building and pre-sold it to low-income people. It was a scam – there were only 500 apartments, and they were pre-sold to thousands of people! The actual building that was under construction was only a show building as well, with no intention of finishing it. The buyers only realized the scam when the scammer ran away. Naturally, they were very angry at losing most of their savings, and they started protesting. Soon, the government got involved and decided to get a contractor and fund a project to build apartments for everyone who had pre-bought them.

A company I was working with got the project. My first concern was how to work with thousands of angry stakeholders! It doesn't matter how well we would have worked – the project was already too late for them, and there was always the chance that they would accuse us of being another scam outfit, even though we were a reputable company working under the supervision of the judicial system in that project. So, we built a website where buyers could look up their apartment and see its up-to-date progress. There was also a room at the entrance with someone there to provide buyers who came over or called with information, because some of them weren't comfortable using the Internet. And finally, there were open site visits once a month.

Working well is not enough; let others know how well you work!

2.1.3.3. Stakeholder engagement

According to NUPP (Nearly Universal Principles of Projects), everything we do

should have a purpose. So, what's the purpose of engaging stakeholders?

There are multiple purposes:

- We want to know all the requirements before it's too late. (Think of the shelter story).
- We want to protect our own goals/reputation/lives/etc. (Think of the thousands of angry buyers in the previous example.)
- We want to benefit from their support.
- Etc.

Some stakeholders have a positive attitude toward the project, which is great, but we still need to be careful and keep it that way. Some have a negative attitude that we may want to change if it's worth the effort.

Once, we were working on an organizational change program and I heard that one of the executives in the company was against us. I was new there and didn't know much about the people, but I was told that the person objecting to our work was always negative and that we should pay no head. I simply couldn't ignore it, though! I asked him for a meeting. At the beginning of the meeting, I told him that we were creating a list of everything that could go wrong with the project, but that I wasn't sure if we'd thought about every risk, and I was wondering whether he could give us more ideas.

He told me about everything he thought could go wrong. I took notes without explaining to him why it wouldn't be so, and every once in a while I asked him to elaborate on a few points. At the end, I thanked him and went away.

A few days later, I went back and told him that we've worked on a few responses to the list of risks, and I'd like to know his opinion about the effectiveness of those. This time, everything switched, and he was giving me various ideas on how to mitigate various risks.

His input was useful to the program, but on top of that, he gradually became a champion of our program, and because of his organizational power, he could bring more resources to the program and help with our success.

2.1.3.4. Focused communications

You should make sure you frequently engage the stakeholders. Don't wait for them to come to you, because that's probably too late. A helpful approach is that of the P3.express methodology, with its monthly stakeholder evaluations, which can be the trigger to various types of engagement.

Another useful element in P3.express that helps with stakeholder engagement is ***focused communication***, whereby specific messages are sent to predefined audiences at different stages of the process to keep them informed of what's going on in the project. This way, if there's any type of feedback, you can receive it as soon as possible and decide what to do with it more easily.

2.1.3.5. Manage by exception

In general, engagement is improved when the person or group is given a level of decision-making power. That's one of the many reasons we want to have proper delegation inside the project. PRINCE2®, a well-known project management methodology, has a nice structure for delegation called ***manage by exception***. In this system, each level has the authority to make decisions up to a certain threshold set by time, cost, quality, risks, and benefits. When it's below the threshold, the higher managers shouldn't get involved, and when it's above that, it should be escalated to the higher levels as an ***exception***.

2.1.4. Focus on value

Let's say you're in the middle of constructing a building that must be finished by a certain deadline to be used for an event. However, your latest forecast shows that you will be two months late, which is completely unacceptable. You have two main options:

1. Lower quality to the minimum acceptable level to make work faster and deliver on time.
2. Spend more money to add resources to the project, make it faster, and deliver on time.

Which one would you choose?

We can't answer the question properly with this information, because we don't

know why we're doing this project! If our goal is to improve our reputation and enter a new market, we can't lower quality because that would defeat the purpose, and so our option is probably to spend more money. On the other hand, if we're doing the project only to earn money, it's probably a good idea to select the first option and lower quality to its minimum acceptable level.

2.1.4.1. Why?!

One thing that is common among all project systems is that we must understand why we are doing the project, because that understanding helps us align ourselves with the goals and increase our chances of meeting them. This is not limited to the project manager – everyone in the team must understand it. This understanding will be used to make important decisions as well. To be clear, the previous example is a high-level decision, and therefore, it should be escalated to the higher levels of the organization to make it anyway.

2.1.4.2. Justification

While all systems agree on the importance of understanding why we are doing the project, they differ on its details. A common way is to focus on the justification for the project, which is one of the principles in PRINCE2® as well. The justification for the project is typically described in a document called the ***business case***, and we continually check it to make sure our investment on the project would be justified by the outcomes and benefits we forecast for it.

2.1.4.3. Value

Instead of ***justification***, the 7th edition of the PMBOK Guide talks about ***value*** (short for ***business value***), and covers justification as a topic below it.

The best way of defining value in this context is the benefits-to-cost ratio. So, if you have two projects and they both benefit you by 1,000 units of money, but one costs you 100 units to do and the other 200, the first one is twice as ***valuable***.

This definition is important because many use the word "value" to mean "benefit" in their day-to-day language, and, for example, talk about "value for money", which doesn't make sense if you consider value to be the benefits-to-cost ratio.

Unfortunately, even some resources about project management mix these two definitions, which causes a lot of trouble.

2.1.4.4. Benefits

So, if we take value to be the benefits-to-cost ratio, cost is relatively simple, and we have to talk more about benefits.

The money you earn from doing something is a common example of benefits. However, benefits are not limited to money; for example, saving lives, improving quality of life, saving the environment, and many other things can be considered as benefits (or at least a quantified form of them, according to some resources). If you improve your reputation by doing a project or enter a new market, that's also a benefit, or at least an outcome that can create benefits.

Different organizations have different ***value drivers***; for example, improving quality of life may be

- the main driver for a public organization,
- a secondary, yet desirable one for an ethical private company that has the primary driver of earning money, or
- of no value at all to some companies.

This is why we say that value and benefits are subjective.

2.1.4.5. Holistic view

Since value is subjective, we can only think about it in the whole context of an organization, and ***not*** within the boundaries of a single project. The value of the product of a project depends on every other project and operation in the organization, both now and in the future, as well as its high-level strategies, and even sometimes politics and regulations in the country or the world. The output of your project may have a certain value in your original setup, but a while later when you create another product with a new project, they can empower each other and increase the value of the first product. Therefore, even those possibilities for the future should be considered when evaluating the project.

Many years ago, I decided to stop working with publishers and self-publish my

work as ebooks. It was a good decision, and my benefits from books increased about 20 fold. I even started buying back the publishing rights of my old books from the publishers and selling them as ebooks on my own website. In case you're wondering, I'm not doing it anymore, because my ***opportunity cost*** has increased. Nevertheless, something interesting I had realized was that each time I published a new book, it increased the sales of the other books by 5 to 15%, which is a great example of the complex, holistic nature of value.

2.1.4.6. Short-term vs. long-term

Here's another thought experiment: You have enough resources to do only one of two potential projects that require the same investment. One gives you a one-time benefit of 1,000 units and the other gives you 300 units per year for the next 20 years. Which one would you choose?

Again, the answer depends on the organization. If you've invested in too many projects with long-term returns, you may prefer the first option to enable you to manage your expenses; or if your company is close to being bankrupt, you would definitely go for the first option. Otherwise, though, if you can afford a long-term return, you would prefer the second option.

2.1.4.7. Portfolio management

So, where should we examine value and decide about it? The answer is in the ***portfolio management*** layer. That's where we select and prioritize the best projects: those that bring us the most value.

Now, you may be asking, if value management is mainly done in the portfolio layer, why do we have a principle about it in the PMBOK Guide, which is about projects? The answer is that while it's mainly done elsewhere, we should still be aware of it in the project layer and align our activities with it on the one hand, and to provide them with up-to-date information they can use to re-evaluate projects on the other.

2.1.5. Recognize, evaluate, and respond to system interactions

You've been given the chance to select one of the following two games and play it just once. In both cases, you will pick a coin from your own pocket (to show there's no trick involved) and toss it, then you have two options:

- **Game A:** If you toss heads, you'll receive €200. If you toss tails, no money will be exchanged.
- **Game B:** If you toss heads, you'll receive €600. If you toss tails, though, you have to pay €200.

Which one would you choose?

2.1.5.1. Loss aversion

The previous example is a problem I've described in a free risk management awareness course, and from about 3,000 people who have taken the course to date, about 78% of them selected the first game. Is that the best choice?

Let's see what the ***expected value*** of each option is:

- **Game A:** 50% × €200 + 50% × -€0 = €100
- **Game B:** 50% × €600 + 50% × -€200 = €200

The expected value of the second option is twice as much as the first one. This means that if we play the games thousands of times, we can have a good level of confidence that the output of the second game would be about twice as much as that of the first one.

It's relatively simple to think of these games when they are played thousands of times, and I think not many would object to selecting the second option in that case. The problem is that I said we're going to play this game only once. So how should we reason about it when it's played only once?

The answer is simple: This game is played once, but we're playing thousands of games like this in our work and our life. If our strategy is to pick the higher expected value, we may lose some of them, but overall, we'd be the winners. There is, however, an exception to this strategy: Picking the higher expected value is a good idea if it fits within your risk appetite. In other words, if all you have is €200, you should go for the first option, no matter what the expected

value of the second option is. This is because the utility of money is not linear.

2.1.5.2. Cognitive biases

Given all the explanation above, some people still ***feel*** more comfortable with the first option. That's natural, because of a cognitive bias called ***loss aversion***. We've evolved to assign more weight to losing than gaining, which was a good strategy for when our ancestors were living in jungles and caves among lions and other dangerous animals, but not so in the modern world. The ratio depends on personal preferences and cultures, but it's usually by a factor of about 2.5. (Losing is 2.5 times heavier for us than gaining.) That's why the 2.0 ratio of the previous example is not enough to urge most people's subconscious to accept the second option.

Loss aversion is innate in us, and there may not be much we can do about it, but we can learn about these cognitive biases and control them consciously instead of leaving the decisions to our ***gut feelings***. Confirmation bias, sunk cost fallacy, effort justification, halo effect, illusion of control, planning fallacy, base rate fallacy, zero risk bias, and stereotyping are just a few examples of the many biases and fallacies that impact us in projects. If you're interested, you can check the ***list of cognitive biases*** on Wikipedia and read more about them there.

2.1.5.3. Critical thinking

Critical thinking is a relatively well-developed domain focused on decision making and the impact of biases and fallacies, with lots of great resources you can use to learn. It's truly essential to any managerial work, including project management, but it can be useful in any aspect of your life as well.

We can see critical thinking as one aspect of ***systems thinking***, which is the subject of this principle. At least, this is the categorization used in the PMBOK Guide.

2.1.5.4. Systems thinking

In general, systems thinking is about reasoning about complexity in a holistic way. If you think about our exploration of ***value*** in the previous principle, our approach

was a holistic one, as expected in systems thinking.

Systems thinking is a wide and relatively complicated topic because of the non-linear interactions within systems. Its most important aspects in project management, in my opinion, are these two:

- Be a critical thinker.
- Think holistically.

2.1.6. Demonstrate leadership behaviors

There was a middle-aged lady in one of the companies I used to work for: a very well-mannered, kind, grandmotherly figure whom everyone liked and respected. Officially, she was a secretary, but unofficially, she was the engine that was running the whole company!

One day, she resigned, because despite her commitment and contribution, it was many years since she'd received a pay raise, and there were companies willing to pay her almost double. The management weren't too bothered; after all, she was only a secretary.

It was only after she left that everyone learned who she really was. Lots of things that seemed to be happening automatically and smoothly were blocked after she left. They gradually added more and more people to fill in her shoes, and at one time, there were five people trying to do her job, and that was still nowhere near enough.

She was helping and guiding people without anyone expecting her to and was caring about things no one thought needed caring about. She was a true ***leader***.

2.1.6.1. Hidden leaders

Many consider ***leadership*** as something limited to managers, but in modern approaches, anyone who influences individuals toward desired outcomes is considered to be a leader. So, why is it important for us?

There are certain people who have the potential and interest to be leaders regardless of their official position. Unfortunately, many overlook these people,

whereas if they paid attention and got to know them, they could get better results by supporting and empowering them; after all, they're getting a free service just because that's someone's way of self-actualization! At the very least, if you don't do anything to support them, don't push them to resign by denying them a small pay rise or a simple verbal appreciation.

2.1.6.2. You as a leader

Identifying and supporting the hidden leaders is one of the things you have to do. The other is that you should improve your own leadership skills.

For example, as a leader, you should know how to motivate people. Let's say someone has done something great in the project, and you want to motivate them by showing your appreciation. How about paying them a bonus? Is that a good idea?

The answer depends on the amount of money you're paying, and the person's need for money. If the amount-to-need ratio is high enough, it would probably motivate them. However, if the amount is relatively low, it would demotivate them. Now, let's say your budget is really limited and you can only spend €100. Instead of giving them €100, which could be offensive to some people, you can order a nice pen and have their name engraved on it. Give them the pen and tell them how much you appreciate them, and that it's a small gift to remind them of the company.

There are many misunderstandings about leadership skills. For example, charisma can help a leader achieve goals, and many think they have to look tough and self-assured to be charismatic. However, there are numerous different forms of charisma, and many of them rely on knowledge, kindness, supportiveness, etc. You can simply check to see which one is closer to your natural self and empower that instead of trying to imitate what charismatic people look like in movies.

2.1.6.3. Leadership skills

What skills does a leader need?

You can find lists of required skills in various resources. APMbok® (another project management guide similar to the PMBOK Guide), for example, categorizes them as follows:

- Communication
- Conflict management
- Delegation
- Influencing
- Leadership
- Negotiation
- Teamwork

There are reputable, reliable resources for each area above as well as many more. You can complete the list, and then start targeting each area and improving your skills in that area by reading books and practicing in the real world.

2.1.7. Tailor based on context

There was a medium-sized company that wanted to enter a new market. To do so, they hired an experienced, successful project manager who had worked for a long time in a large organization well known in that market. After coming to the new company, he started implementing the project management system they used to have in the large company. It was a complete failure!

There were two main problems:

1. He was trying to implement a copy of something that was customized for a company with a size, culture, and background all different from the current company.
2. He was trying to implement a complicated system in one go, whereas it had probably been formed gradually and organically in the previous company.

The first issue is the subject of this principle: Project management systems should be tailored to match their environment.

2.1.7.1. When it's too expensive

Have you ever heard something like, “No, we don’t use PRINCE2® because we’re a small company and it’s too expensive for us.”?

It would be too expensive if it wasn’t tailored to suit the company. When tailored properly, though, it should help you save costs, improve your reputation, and more.

So, how do we tailor a project management system?

As an example, in PRINCE2, team leaders should send ***checkpoint reports*** to the project manager at predefined intervals. PRINCE2 calls that a ***management product*** rather than a document, simply because it doesn’t have to be a document. If it’s a large or complicated project, you may decide to have formal, written documents, but if not, even a phone call is enough. The only thing that’s necessary is to have an understanding that those phone calls should exist at certain intervals and the required information communicated. When you decide about how you want to form your checkpoint reports in a methodology like PRINCE2, you’re tailoring your system.

2.1.7.2. The tailoring process

There are various ways of tailoring a system, and there’s more information about it in the coming chapters. For now, let’s focus on the tailoring process.

PMBOK 7 recommends a two-step tailoring process: first, a high-level, partial tailoring for all projects in the organization, and then a final tailoring for each project. This is so because usually there’s a lot in common among projects being done in a single organization. So, instead of leaving the whole tailoring effort to individual projects, we can simplify the process and make projects more uniform by doing the high-level tailoring at the organization level.

On the other hand, each project has its own environment, so we can’t fully tailor the systems at the organization level. That’s why the first step is only partial, and then we leave the rest to the individual projects.

2.1.7.3. The difficulty of tailoring

To be honest, tailoring is complicated! Some people simply start cherry-picking

from various project management systems and call it tailoring, whereas what they create in this way won't have internal consistency and usually doesn't work well. In many cases, it resembles Frankenstein's monster! In general, when tailoring, we're building a new system, and like any other system, we must take a holistic, critical view.

Most systems leave the responsibility of tailoring to the project manager. However, someone who is an otherwise great project manager may not necessarily have all the skills required for building a system. It's simply unrealistic to expect every project manager to have such a range of skills. Something interesting in DSDM® (a project management methodology built for Agile projects) is that they've identified this problem and assigned this responsibility and a few others to a separate role they've called the ***DSDM Coach***.

2.1.7.4. When to tailor

Another concern in tailoring is timing. Should you tailor your system at the beginning, or rather start using the system and then gradually tailor it? The answer depends on the methodology you're going to use:

- Maximalist systems such as PRINCE2® have many elements that cover every need in almost any project. That's why you can't use them as is in most projects without tailoring them at the beginning.
- Minimalist systems such as P3.express only contain the essentials that are in common to most projects. Therefore, you can start using them without upfront tailoring, and only start a gradual tailoring after a while to enrich them.

You must always be careful with upfront tailoring because it's common to come up with a fancy tailored system that later on proves to be too formal or heavy for the project. Always keep the upfront tailoring to a minimum and allow the rest to happen gradually by adapting to your environment.

2.1.8. Build quality into processes and deliverables

Have you heard people saying that quality management is too expensive?

That's another example of poor tailoring. You have to expend some effort on your

quality management, and in return, you'll save extra effort by reducing reworking, and improve your reputation as well. There should be a balance here, and what you get out of the system should be more than what you invest in it. If not, your quality management system is not tailored properly.

2.1.8.1. Testing

For some people, quality management is limited to testing the output and making sure it's fine. While various forms of testing are necessary as a final step, that's by no means enough. The main concern in quality management is to prevent poor results rather than finding and fixing them. That prevention is how we can save time and money. It's about being proactive, which happens to be one of the NUPP principles as well.

2.1.8.2. Not an afterthought

When quality management is an afterthought that's only bolted onto the system, it won't be effective enough. Instead, it should be built into the system and integrated with everything else.

There are two elements to quality management, and your system should cover both:

- The quality of the product and its production processes
- The quality of the project management system

2.1.8.3. Quality of the technical work

Managing the quality of the technical aspects depends on the type of product, as shown in these examples:

- **In an IT development project**, you can have automated and manual tests to make sure your output works fine, and also use practices such as ***pair programming*** to be proactive and increase the likelihood of the output passing the tests. Pair programming is when two programmers pair up and work together for a day, during which time one programmer writes code while the other observes and comments, and every half an hour or so, they switch

places. It helps with quality, and it also helps people learn from each other.
- **In a construction project** with a concrete structure, you would have different quality methods. There are various types of non-destructive and destructive tests for concrete, but on top of that, there are guidelines about the type of cement, sand, and water you may use, and the way you should mix and cure the concrete to make sure it will have the expected properties.

So, as a project manager, how would you know about all these technical aspects?

As usual, you don't have to know about them all yourself, and that's not a problem because you have a team of experts who should know it. What you have to do is to make sure the quality methods are identified, documented, and executed. Does your customer want to approve the quality method? Does your own organization have extra quality expectations for all projects? Are there national codes you must follow in the project? You should find answers to all these and other questions and make sure they are answered satisfactorily.

2.1.8.4. Quality of the managerial work

You should pay attention to the quality of the project management work as well; but how would you do that?

A good example is the approach in P3.express, where you need to have ***peer reviews*** at certain intervals. For each peer review, you'd ask another project manager to review your work and give you comments. It's recommended to ask a different person for each peer review to benefit more from a diversity of ideas.

Peer reviews help you increase the quality of your work, but they also help both parties learn from each other. It's a great opportunity you shouldn't miss regardless of which project management methodology you use.

2.1.9. Optimize risk responses

One of the companies I was working for had multiple construction projects, and I used to visit each project once a week to evaluate its progress. In one of the projects, there was a nice worker in the concrete team. Every time I was there, he

used to shout, “Hey, welcome back, sir!” with a big smile, and we used to exchange a few pleasantries. He had a family in a city far away, and he was always looking forward to holidays to go back and stay with them for a few days.

One day, shortly after I arrived at the site, I noticed people running toward somewhere. I went there as well, and I was met by the worst scene in my whole life: He had been working on the fourth floor, where there were still no walls, and he had fallen off the edge. He had fallen into a pile of steel bars that were folded for reinforcing the concrete. Just like in a horror movie, the bars had punctured him in multiple places, and his dead body lay in a pool of blood.

2.1.9.1. Risks

The tragedy above is an example of uncertainty: When there’s no wall to a floor and people have to work there, someone ***may*** fall down.

Any uncertainty that can impact the project, either in a negative or a positive way, is called a ***risk***. These are not certain like the fact that we should build the wall, and so they require a different mode of management.

2.1.9.2. Identifying risks

First, we should identify the risks. We can have workshops at the beginning of the project with all key team members and ask them to give ideas on various uncertainties in the project. However, we can’t identify all the risks at the beginning, and some can only be identified when we become more aware of them by getting closer to them. On the other hand, sometimes the environment changes and new risks are introduced. That’s why we should be continuously looking out for them. A good practice is to have a short, fixed section at the end of most meetings to explore whether new risks can be identified.

2.1.9.3. Responding to risks

After identifying risks, we should start planning responses to them. What do you do when you know people need to work on floors that don’t have walls yet and someone may fall off?

A simple response is to add a new activity to the project to build temporary handrails as soon as the floor is built. That's a good idea, but is that enough? Maybe we can do better. Can we limit what is done on those floors until the permanent walls are built? That requires making adjustments to our plan. While we do all of these, there may still be some unplanned, unnecessary traffic on those floors. To cope with that, we can have a health and safety training program for everyone. This can be a response to multiple related risks, which makes it even more interesting.

No matter what you do, there may be some accidents. If you have a large project, you can set up a first aid center with a nurse to reduce the immediate impact of injuries.

We have a social responsibility to make sure people are safe in the project. When we take all the reasonable actions, there's still a chance that something will happen. The sad reality is that we can't prevent all tragedies. So, the last step is to protect the project with a proper insurance policy so that if the tragedy does happen, at least there won't be a big financial setback for the project.

What's explained above is ***risk management***: taking actions before uncertainties happen in order to reduce their probability or impact. Having proper risk management helps us reduce costs, improve our reputation, respect our social responsibility, and make the project more predictable.

2.1.10. Navigate complexity

In your opinion, what's the biggest source of complexity in projects?

I don't know what you have in mind, but in my opinion, the biggest source of complexity is that there are human beings involved in projects! People are complicated, and when working together, they create a complex environment. We've already talked about this in the first three principles.

Non-linear relationships are another source of complexity that require a holistic view along with critical thinking, which we've also talked about in the "recognize, evaluate, and respond to system interactions" principle.

Another source of complexity is ambiguity and uncertainty, which we also talked

about in the previous principle.

My opinion is that all the other principles are various ways of navigating complexity, and we don't need to have a separate principle for it. However, the rest of the team believed it's best to have a separate principle to emphasize it. So, here I am, writing a book about PMBOK 7, reaching this principle without having anything to add after the previous principles!

2.1.11. Embrace adaptability and resiliency

What is Agile?

Most people give vague or meaningless answers to this simple question, especially those who claim to be Agile experts or Agile practitioners. So, let's talk about it.

2.1.11.1. Adaptive vs. predictive

There are two ways of going about anything:

- **Predictive**: You think about every aspect of the output upfront, design it carefully, and then build it. Think of sending a rover to Mars or building a bridge; that's the only reasonable way you can go about them.
- **Adaptive**: We can't predict everything when the way output works depends massively on the acceptance of the end users, which is mainly a problem for software applications. In that case, instead of trying to predict everything, which we know doesn't work well, we can create an environment in which we can build subsets of the output, have end-user representatives work with it, collect direct and indirect feedback from their work, and use that to decide how to proceed in each consecutive step.

Adaptive methods are commonly called ***Agile***. The Agile community usually calls predictive systems ***waterfall*** and ***traditional***, but we don't refer to them as such in the PMBOK Guide, because those words can have a negative connotation. Both approaches are valid, though, and each works well for a specific type of output.

2.1.11.2. PMBOK and Agile

The previous versions of the PMBOK Guide were mainly created for predictive projects. As adaptive methods became more popular, more information about adaptive systems was bolted onto the PMBOK Guide, but it was never truly integrated. Since the 7th edition was written from scratch, we had the opportunity to make it truly compatible with both approaches.

Some people say that the 7th edition “has become Agile”, which is ***wrong***! It hasn’t “become Agile”, but it has become fully compatible with both approaches, or has become approach-agnostic, if you will.

2.1.11.3. Adaptability as a principle

So, how come there’s a principle about adaptability when the PMBOK Guide is supposed to be compatible with and recognize both approaches?

The reason is that we have two subjects:

- Adaptability for the product we build in the project
- Adaptability for the project management system

You can build the product using an adaptive or a predictive approach depending on the type of product. However, for the project management system itself, since it’s about complex relations among human beings, you need to have an adaptive system. It means that it’s best to have a base system, and gradually enrich it during the project; or if that’s not possible, at least revise the system to suit the environment as you progress.

2.1.12. Enable change to achieve the envisioned future state

Imagine you work in a government agency and have a fixed budget to build a few clinics in remote cities. This budget is enough for building 10 clinics; however, if the projects are managed properly, it may be possible to build 11 of them. That one extra clinic would exist because of good project management, and any lives saved because of that could be partially attributed to proper project management.

Isn't that great?

Many projects are undertaken around the world to educate children in low-income countries, to help various groups of people to build a proper life, to find cures for diseases, to gain new scientific achievements, etc. All of these are the elements that make the world a better place for everyone.

Unfortunately, many have focused too much on ***fancy*** projects in recent years. While entertainment is necessary as well, it's not the biggest concern, and what is done in those areas cannot be easily expanded to all domains.

2.1.12.1. Change vs. outcome

Projects are about structured changes. Although, when we talk about ***change***, what most people think of is ***outcome***, and they mix these two. For example, if you implement a document management system in your company, the ***output*** of the project would be the implemented system, which is a ***change*** in state (there was no document management system before). However, there's a difference between having such a change and having an ***outcome*** such as easier and more efficient access to documents. The latter is what we're looking for, and it may sometimes be called a ***change***, which causes a lot of confusion.

2.1.12.2. Managing outcomes

So, since "change" normally means achieving a desired "outcome", let's talk about outcomes.

What we build in a project is the ***output*** (a product). The output has the potential to create ***outcomes*** in the wider context, but in many cases, this potential depends on extra activities (e.g., training) or projects. That's why managing ***outcomes*** is primarily the subject of ***program management***.

Similarly to what we had before about justification and value, the fact that something has to be done at a higher level doesn't mean we don't have anything to do with it in the project. We still have to understand why we're doing the project and what the changes and desired outcomes are, and align ourselves with them. We also have to provide the higher levels with the information they need to

manage those outcomes.

So, as a project manager, feel free to see yourself as a change enabler.

2.2. NUPP principles

Now that we've explored the PMBOK 7 principles, let's take a look at an alternative resource with a similar purpose: NUPP. NUPP stands for Nearly Universal Principles of Projects. It's an open (non-proprietary) resource you can use to re-interpret other systems with more efficiency.

These are the NUPP principles:

1. Prefer results and the truth to affiliations.
2. Preserve and optimize energy and resources.
3. Always be proactive.
4. Remember that a chain is only as strong as its weakest link.
5. Don't do anything without a clear purpose.
6. Use repeatable elements.

So let's have a brief look at these principles. The introduction to each principle is quoted directly from NUPP (https://nupp.guide), which is fine, because it's an open resource and everyone's allowed to use it for commercial and non-commercial purposes; i.e., there's no copyright problem!

2.2.1. Prefer results and the truth to affiliations

> We all have a natural tendency to belong to groups, a tendency that often goes beyond its basic form, creates strong affiliations, and causes problems. We lose a lot more than we gain because of affiliations. We can become more professional and effective experts if we don't limit our identity and preferences to certain groups.

For example, some practitioners turn their method into a religion and start objecting to everything else. As a true practitioner, you shouldn't care about those childish or cult-like behaviors, and simply learn about all the existing resources, and use the most appropriate one(s). Even for those systems you don't use, you

can still be inspired by and use the knowledge to enrich your selected method.

2.2.2. Preserve and optimize energy and resources

> Resources are limited. Resources available to the project are limited, as is the mental energy you have to make good decisions. You should preserve and optimize this resource for yourself and for the project, and help other team members do the same.

For example, you can use the ***80/20 rule***, also known as ***Pareto law***, in your project management system: Instead of trying to achieve 100% of the potential benefits of structured project management with 100% of effort, target 80% of the benefits, which may take only 20% of the effort.

You should also be aware of ***decision fatigue***: If you spend too much of your energy micromanaging people in the project, you will be left with less energy to spend on your true responsibilities.

2.2.3. Always be proactive

> There's a natural tendency in us to be reactive. It can help us preserve our energy by not dealing with unimportant matters, or it may give us better results when we are dealing with something about which we're completely incompetent. Those situations are different from our projects, and here we can get better results by being proactive.

Risk management is a simple example of being proactive.

2.2.4. Remember that a chain is only as strong as its weakest link

> There are various domains in projects, and they all need attention; we must have a holistic perspective of the project. Paying attention to a seemingly important domain (e.g., time) is not enough, because all domains interact and they don't work properly unless they all receive adequate attention.

It’s common for project managers to spend too much time scheduling the project and yet not spend enough time on other aspects of management, which in turn makes the scheduling almost useless. We must pay attention to every aspect.

2.2.5. Don't do anything without a clear purpose

> You shouldn’t do anything unless it has a clear purpose. Imagine two parallel worlds where everything is the same except for the thing that you’re considering doing: How different would those worlds be? Is the difference worth the effort to do that thing?
>
> If you don’t have a clear purpose in mind and are only doing something because everyone else is doing it, or everyone says that it’s important to do it, then
>
> - it may not have a real benefit in your case, and therefore you may not get anything out of it; or
> - it may still have potential benefits in your case, but because you don’t have the purpose in mind, your way of doing it may not help realize the benefits.

For example, PRINCE2® says you need to have something it calls a ***highlights report***. You can download a template from the Web and fill it in every time and send it to the people it tells you to. Would that be enough, though, or does it remind you of the ***Cargo Cult*** we talked about at the beginning?

Instead, you can read PRINCE2 to understand what the purpose of a ***highlight report*** is, and if you believe it makes sense to your project, create a document that can serve that purpose. When you’re done, take a look at the templates, and if there’s a difference, consider why it is so. If you can’t see the reason why, simply ignore it for now.

2.2.6. Use repeatable elements

> An ad hoc approach to the project takes too much energy and resources, and always runs the risk of missing some of the necessary elements. The best way of simplifying what has to be done is to use repeatable

elements, and preferably to take them in repeatable cycles.

A good example is quality checklists. If there's something you have to check frequently, create a simple checklist that contains everything you have to check. This way, you don't have to think about it every time and you can spend all your mental energy on the actual work. As you use the checklist, you will find ideas for improving it, which is a great form of continuous improvement.

Another example of repeatable elements is the project management cycles used in P3.express and some other systems. You have a cyclic approach that repeats constantly, and it gradually becomes second nature to you without requiring too much of your mental energy.

2.3. PRINCE2® principles

PRINCE2 is a well-structured project management methodology, and we'll talk more about it in the next chapter. It also has a set of principles:

1. Continued business justification
2. Learn from experience
3. Defined roles and responsibilities
4. Manage by stages
5. Manage by exception
6. Focus on products
7. Tailor to suit the project

These principles more or less apply to all projects, but they are specifically designed to support the PRINCE2 methodology rather than being a guide for every project.

2.3.1. Continued business justification

Hopefully, we only start projects that are justifiable. However, that's not enough, because things change and a project that used to be justifiable may lose its justification along the way. Unfortunately, most of these projects are continued without paying attention to this issue, sometimes because of being blind to it, and sometimes because of being victims of the ***sunk cost fallacy***.

We must frequently check the justification of the project. A good way is to have certain, predefined, frequent points when we stop and ask whether or not we should continue the project based on the latest information. Even if your project doesn't lose its justification and you continue it, these gates remind you of why you are doing the project and help keep everything aligned with the goals.

2.3.2. Learn from experience

It's too expensive to reinvent the wheel all the time, and that's why we prefer to use the existing project management systems such as PRINCE2®, P3.express, and the PMBOK® Guide instead of building everything from scratch.

On a different level, inside projects, we can significantly benefit from the lessons learned in previous projects, especially those that were similar to the current one. We can do this when we have a proper way of documenting and sharing lessons.

2.3.3. Defined roles and responsibilities

Except for in really small projects, it becomes very difficult for people to work together unless they know what is expected of them and what they can expect from others. That's why we need to have defined roles and responsibilities.

2.3.4. Manage by stages

It's not realistic to plan a large project with all the details upfront. There's only a small window of time in front of us within which we can create a detailed plan. So, it's best to have a high-level plan upfront, and then add details for each upcoming stage just before it begins.

2.3.5. Manage by exception

We must have proper delegation in projects, to use the decision making power of more people, create buy-in, and save energy. To do so, we set tolerances for each level of management, and if the consequences of a decision are below the threshold, we let them decide without interfering, but if they're above that, we

expect them to escalate it to the higher level.

2.3.6. Focus on products

Many of us are focused on activities, but that’s not a good idea because each set of activities implies a certain solution, whereas we usually have more than one solution for each problem. So, instead of thinking about activities all the time, we prefer to be focused on the products (the outputs of projects), to examine various ways of creating them, and to consciously select the best set of activities based on that.

2.3.7. Tailor to suit the project

Most systems, especially PRINCE2®, cannot be used without proper tailoring. Moreover, we need to have continuous adaptation and tailoring for the project management system, both for those systems that should be tailored upfront and those like P3.express that don’t require initial tailoring.

3. Selecting a Methodology

The PMBOK Guide isn't a methodology; i.e., it doesn't give you a path or flowchart you can follow in your project. Instead,

- the principles help you understand and interpret everything more effectively (the topic of the previous chapter), and
- the ***performance domains*** help you enrich the methodology (the subject of the next chapter).

So, to embrace the whole context, we're going to have a brief look at a few methodologies in this chapter. P3.express, PRINCE2®, DSDM®, and Scrum are examples of resources that show you the way. To avoid confusion, it will be helpful to note a few things:

- The word "methodology" is a taboo in some Agile communities, and they prefer to use other words instead. For example, Scrum calls itself a ***framework***. The word "framework" has a wider meaning, though, and can be used for other purposes as well.
- Some resources such as the European Commission's PM² call themselves methodologies, whereas they are more like guides.
- P3.express tries to avoid unnecessary arguments and simply calls itself a ***system***, which is probably the most general word that can be used.

Regardless of these disagreements over labels, we'll focus on systems that show the way, and for simplicity, we'll call them ***methodologies***, which is the commonest word for describing this purpose.

Finally, note that these systems are not the same; for example,

- Scrum is designed for small projects with only a few team members. If you have a bigger project, you can't use it without heavy tailoring, which may not be worth the effort, and it would be easier to use a method designed for larger projects.
- Scrum and DSDM are designed for IT development projects. If you have a different type of project, you would need to heavily tailor them, and again, it would be easier to use one of the other methods that are designed to be

general and usable in every type of project.
- Scrum covers a subset of project management concepts rather than the whole subject. This can be fine in simple projects, but others need to cover all concerns. Adding the remaining elements to Scrum is possible, but it may be easier to start with a system that already has all of them.

So, let's take a look at these systems. Note that the goal of this chapter is not to learn how each of these systems works, as that's a much lengthier topic. The goal is, rather, to gain a high-level familiarity with these methods to help you understand the role of the PMBOK Guide and especially the topics of the next chapter.

3.1. P3.express

P3.express is an open (non-proprietary), minimalist project management methodology usable in most small, medium, and large projects. It's not designed for micro-projects or mega-projects; for micro-projects, there's the micro.P3.express variation, and for mega-projects, practitioners would need to heavily tailor the system before using it, or use a methodology designed for mega-projects.

P3.express is designed to be used without upfront tailoring and only recommends gradual, ongoing tailoring during its use.

3.1.1. Team structure

The following are the roles in P3.express:

- **Sponsor:** They are senior business-oriented people who make high-level decisions for projects and provide resources.
- **Project Manager:** They are responsible for the day-to-day management of projects.
- **Team Leaders:** They lead the internal teams (if any) and are in touch with the project manager.
- **Supplier Project Managers:** They lead the external teams (if any) and are in touch with the project manager.

You are allowed to introduce new roles, but you should only do it gradually and based on the feedback you collect from your environment, and without disturbing the simplicity and minimalism of the system.

3.1.2. Process

A P3.express project starts with ***project initiation*** and ends with ***project closure***. Between those two, there are monthly cycles where each starts with a ***monthly initiation*** and ends with a ***monthly closure***. During each month, there are certain ***weekly management*** and ***daily management*** activities. Finally, after the project ends, there's a ***post-project management*** cycle for evaluating the benefits of the project.

We start by initiating the project. That's where we appoint the key team members and create a high-level plan. The information is then sent to the ***sponsor*** to decide whether or not to start executing the project.

The upfront plan is preferably high-level, and each ***monthly initiation*** revises the high-level plan and details it for the upcoming month. Toward the end of ***monthly initiation***, the updated information will be sent to the ***sponsor*** again to decide whether or not to continue the project.

We work during the month, and each week we measure performance and respond to deviations. There are also daily activities for accepting completed deliverables and managing follow-up items (issues, risks, change requests, improvement plans, lessons learned).

At the end of the month, the ***monthly closure*** activities evaluate stakeholder satisfaction and create new improvement plans based on them.

When everything is done, or we decide to prematurely stop the project, the ***project closure*** activities will tie up the loose ends, evaluate stakeholder satisfaction for the last time, hand over the product, archive the documents, etc.

The ***post-project management*** cycles repeat every 3 to 6 months for 1 to 5 years, depending on the project, to evaluate its benefits and come up with extra actions that can increase benefits, or ideas for new projects.

Each of the seven management activity groups mentioned above (except for ***daily management***) ends with a ***focused communication*** activity, where a short message with defined content is sent to relevant stakeholders to make sure everyone stays informed.

The ***project initiation*** and ***monthly initiations*** have a ***peer review*** activity that plays a key role in P3.express: When you're almost done with the work, you'll ask another project manager in the company to come, check your work, and give you feedback. It's preferable to get help from a different project manager each time, if possible. These peer reviews help both parties learn from each other, and they are also an effective way of finding and fixing problems before they pile up.

There are four artifacts in P3.express:

- **Project Description:** It contains basic information such as the reason for doing the project, its budget and time, stakeholders, etc.
- **Deliverables Map:** It's a breakdown of the product into its building elements (deliverables), preferably created as a mind map. Each deliverable needs to have a ***custodian*** to watch its status and report on it.
- **Follow-Up Register:** It's a list of all risks, issues, change requests, improvement plans, and lessons learned. Each item is assigned to a person as its ***custodian***, who follows up on it until it's closed.
- **Health Register:** It's a document that stores the results of peer reviews and stakeholder satisfaction evaluations.

3.2. PRINCE2®

PRINCE2 is a well-structured, general-purpose, sophisticated project management methodology. You're not supposed to use it without upfront tailoring. What comes out of the box is more compatible with large to super-large projects, which means that it requires more tailoring effort for medium and small projects.

3.2.1. Team structure

PRINCE2® recognizes three levels of management inside the project, connected to another level above them in the organization. There are various roles defined in

each layer, including but not limited to the following:

- **Directing layer**
  - **Project Board**
    - **Executive:** They are business-oriented people responsible for the outcomes of projects and for achieving the goals. They are high-level managers in the organization and make the high-level decisions in the project.
    - **Senior User(s):** They are one or more people who understand the needs of users and bring that knowledge to the Project Board.
    - **Senior Supplier(s):** They are one or more people who understand the suppliers and teams and bring that technical knowledge to the Project Board.
- **Managing layer**
  - **Project Manager:** They are responsible for the day-to-day management of the projects and creating their outputs within the set targets of time, cost, quality, overall risk, and benefits.
  - **Project Support:** They help the project managers in their activities.
- **Delivering layer**
  - **Team Manager(s):** They lead the internal and external production teams and are in touch with the project manager for coordination.

You can introduce new roles by splitting the default roles. On the other hand, you can also merge some of the roles to create a simpler setup. There are a few constraints, though; for example, you may not merge the Project Manager and Executive roles, because it would create a conflict of interest, and also, it's difficult for one person to properly do things that are so different in nature.

Various activities in the PRINCE2 manual have tables where they explain the responsibility of each role for that activity.

3.2.2. Process

When the idea of a project has been mooted and ***before*** we make sure it's a good idea to run it, we'd run the ***starting up a project*** process. This process appoints the Executive, Project Manager, and a few other key team members. They create a few artifacts, which are mostly packaged in a key artifact called the ***project***

brief. This contains information about the reason for doing the project, its justification (an outline of the ***business case***), the approach to creating the output, etc.

The Executive and possibly other people in the higher levels of the organization will review the project brief to see whether or not it's a good idea to run the project. If it is, they would allow the project to proceed to the next process, which is the ***initiating a project*** process.

The team spends more time on initiation and creates a complete, yet high-level plan for the project. Most of the findings during this process will be packaged in an artifact called the ***project initiation documentation***, sometimes abbreviated as ***PID***. It contains more precise estimates for time, cost, scope, and other targets, as well as a business case that's refined based on this information and gives us a more reliable picture of the justification of the project.

The project initiation documentation will be sent to the Executive and possibly other decision makers, and this time, they make a final decision whether or not to start executing the project.

As you can see, the decision for executing the project is made in two steps: first a quick, rough study to exclude ideas that are in no way justifiable, and then a longer (more expensive) one to make sure they really are justifiable.

PRINCE2® projects are run one ***stage*** at a time. Before each stage, the ***managing a stage boundary*** process will be run to revise the high-level plan and create a detailed plan for the upcoming stage. The business case will be revised based on the new information and everything will be sent to the Executive to decide whether or not to continue with the project. All high-level decisions of the Executive are made in a process called ***directing a project***.

During each stage, the ***controlling a stage*** process manages the day-to-day work by handing over work plans to Team Leaders and receiving back completed deliverables. There's also a ***managing product delivery*** process in the delivery layer, run by Team Leaders, which interacts with the ***controlling a stage*** process run in the management layer.

The ***controlling a stage*** process also measures progress, issues reports, and looks for deviations. When a deviation is found, if it's within the decision-making

threshold of the project manager, the project manager is expected either to decide how to recover from it or otherwise to escalate it to the higher level and wait for them to decide.

When the project loses justification or when everything is finished, we move to the ***closing a project*** process, where we tie up the loose ends, hand over the product, and close it.

3.3. Scrum

Scrum is one of the first-generation Agile systems designed for IT development projects with a single team of up to 10 team members. In one sense, it was more or less a lightweight management layer built to provide the necessary elements for the commonest Agile system of that time, XP (eXtreme Programming), which was focused on the technical aspects only. However, it gradually evolved from that initial state and became a standalone system.

Scrum provides a well-formed, simple system for some projects and provides them with what it considers to be enough project management. Its positive attributes, along with various environmental factors, have made it the most popular Agile system. Nowadays, most Agile projects either use Scrum or one of the second-generation systems that are highly inspired by Scrum.

3.3.1. Team structure

There are three roles in Scrum:

- **Scrum Master**: A single person responsible for making sure Scrum is used properly, and for solving problems, facilitating, etc. The Scrum Master is focused on the context rather than the content.
- **Product Owner**: A single person with business understanding who's the main hub between the team and the rest of stakeholders. They understand what is expected and bring that understanding to the team. They are also responsible for ordering the work items based on their value, so that what the team produces at any one time has the highest possible value.
- **Developers**: Anyone who does the technical work of the project is called a

"developer"; e.g., programmers, database admins, UI designers, architects, etc. Their main job is obviously creating the product, but they also contribute to the management of the project.

Scrum has a decentralized project management system, meaning that none of these roles is a project manager and the project management responsibilities are distributed among all roles.

Because of the way Scrum is designed, adding new roles will harm its internal consistency, especially if it's a role such as project manager.

3.3.2. Process

Contrary to most systems, there's no initiation or closure in Scrum, which is considered a serious issue by some, and a feature by others. In reality, this is because Scrum targets a subset of project management activities.

At the beginning, the Product Owner talks to various stakeholders and collects a list of things to do. It doesn't have to be a complete list, as more will be discovered as we proceed with the project. The list is called the ***Product Backlog***.

The Product Owner is responsible for composing the items on the Product Backlog and ordering them based on their value, so that the more important ones are on the top.

Scrum has fixed-duration cycles called ***Sprints***. Each project team decides about the fixed duration of their sprints, but it has to be shorter than one month.

Each Sprint starts with a ***Sprint Planning*** meeting (maximum 8 hours), where team members get together and the developers decide how many of the items on top of the Product Backlog they can finish by the end of the Sprint. They move those to a second list called the ***Sprint Backlog***. It's a helpful practice because a large list like the Product Backlog can be overwhelming, whereas a short one like the Sprint Backlog creates focus. They also set a goal for the Sprint and add it to the Sprint Backlog; an overall goal that can be used to better interpret the individual items.

After that, they start working. The Developers create the product, while the Scrum Master is there to help solve their problems if needed, facilitate their meetings, and make sure the framework is properly adhered to. The Product Owner is also available to help explain the items and issues if there are any doubts.

While they are working, they have 15-minute daily meetings called ***Daily Scrums***, where they talk about what they've been doing, what they are going to do, and what problems they may have.

Sprints have a fixed duration and end when the time is over, even if some of the items in the Sprint Backlog are not finished. Those items will return to the Product Backlog and the team will proceed to the closing events of the Sprint. It's essential to only proceed with items that are 100% complete, because that's the only way we can have reliable feedback. For this reason, there's a ***Definition of Done*** that describes (or implies) when an item is truly done. If something doesn't satisfy all the conditions of the Definition of Done, it will be returned to the Product Backlog to be completed later.

The first closing event is a ***Sprint Review***, where the team and the rest of stakeholders spend maximum 4 hours reviewing the latest features and collecting feedback. This is only one of the many opportunities for collecting feedback and not enough for true adaptation. The team has to provide the latest ***Increment*** of the working software to end users or at least end-user representatives to try out and generate feedback.

The last event is the ***Sprint Retrospective***, where the team gets together and spends maximum 3 hours thinking about how they can improve their project in the next Sprint.

3.4. DSDM®

DSDM is another first-generation Agile method. In contrast to Scrum and other systems of the time, it was created from the ground up to be compatible with large, complicated IT development projects with multiple teams. Its structure is inspired by PRINCE2®.

3.4.1. Team structure

DSDM® has an interesting, though relatively complicated team structure:

- **Project Level:** This contains a number of roles responsible for the overall project outputs/outcomes and coordinating the teams.
  - **Business Sponsor:** This is the most senior role in the project, responsible for its justification, budget, etc.
  - **Business Visionary:** They interpret the business needs at the project level.
  - **Project Manager:** They are responsible for the day-to-day management of the project.
  - **Technical Coordinator:** They are responsible for coordinating the technical aspects of the project across teams.
- **Solution Development Team Level:** There can be one or more teams, each with a number of roles required for coordinating themselves with the project level, making decisions within their threshold, and producing the output.
  - **Team Leader**: They are responsible for the management aspects inside the team, and they are in touch with the Project Manager.
  - **Business Ambassador:** They bring their business and user expertise to the team, and they are in touch with the Business Visionary.
  - **Solution Developer:** They are the people who build the product.
  - **Solution Tester:** They help ensure the quality of the product.
- **Support Level:** This contains a number of roles that support multiple teams.
  - **Technical Advisor:** They help teams in certain technical aspects; e.g., security.
  - **Business Advisor:** They help teams by providing them with business and user information; e.g., legal and regulatory aspects
  - **DSDM Coach:** They help teams understand how to use the DSDM methodology.
  - **Workshop Facilitator:** They help everyone by facilitating their workshops.

In addition to the roles above, there's also a ***Business Analyst*** role that belongs at both the project level and the team level. They have both business and technical knowledge and bridge the gaps between them.

3.4.2. Process

A DSDM® project starts with a ***pre-project*** phase to evaluate the project and decide whether or not to invest on it. The output will be stored in an artifact called the ***Terms of Reference***.

If the decision is made to proceed with the project, the ***feasibility*** phase is then run. During this phase, we'll spend more time on the business, technical, and managerial aspects of the project and create a ***feasibility assessment*** artifact, which will be used to decide whether it's a good idea to go even further.

The next phase is ***foundations***, which puts more effort into the same areas targeted during ***feasibility*** to create a foundation for the project (i.e., a high-level plan). The output will be stored in the ***foundations summary*** artifact. If needed, this phase can be repeated in the project to refine the foundation.

After creating the foundation, we proceed with the ***evolutionary development*** phase. We have many cycles of this phase, and for each, we detail the foundations to create a plan for the cycle, and at the end, create a new ***increment*** of the product.

There's also a ***deployment*** phase, which may run with the same frequency as ***evolutionary development***, or a lower one. Each time it's run, it puts the latest ***increment*** into production for the end users. In general, we prefer to have as many deployments as possible, because that makes adaptation easier; however, that can be limited by business requirements, risk of disturbing the users, etc. There's an optimum deployment ratio that depends on the type of product.

A form of closure is implied in the ***deployment*** phase, and therefore, there's no separate closure phase for the whole project. However, there's a ***post-project*** phase for evaluating the benefits of the project after it's finished.

DSDM projects are timeboxed, meaning that the whole project ends exactly as defined upfront, and during that time, we try to deliver as much as possible. DSDM® heavily depends on using the ***MoSCoW prioritization*** technique to support this method.

MoSCoW stands for the following:

- **(M)ust-have items:** Items we must have in the product, because without them the product will be unusable (e.g., security for a banking application).
- **(S)hould-have items:** Items we should have in the product as we'd run into problems without them. However, we can have workarounds for those problems (e.g., doing the task manually).
- **(C)ould-have items:** Items that would be nice to have and would add value, but whose absence won't cause problems (e.g., having a dark theme in the banking app).
- **(W)on't have this time:** Items we decide to exclude.

In the beginning, during the ***feasibility*** and ***foundations*** phases, we create a list of items along with their priorities and order them. The list is called the ***prioritized requirements list***. The items are high-level at the beginning, and we break them down into smaller items as we proceed, especially when detailing the plan for development iterations.

To have enough flexibility in the project, we should have no more than 60% must-have items and no less than 20% could-have items. When estimating the fixed duration of the project,

- we must have enough time to finish everything in an optimistic way, and yet
- we should be able to deliver all the must-have items within our most pessimistic estimates.

When working on the project, our progress measurement will be focused on checking how many items we can finish by the end of the project. If, at any time, we forecast that we can't deliver all the must-have items, we should probably cancel the project, or at least make fundamental changes.

Basically, we guarantee that we'll deliver all the must-have items, we'll do our best to deliver all the should-have items, and we'll see what we can do about the could-have items.

4. Enriching the Methodology

We've had a quick review of a few methodologies in the previous chapter: systems that give you a path through the project. When you have such a system, you can proceed and enrich it using the PMBOK Guide and other project management guides such as APMBoK and PM², or using various books that focus on different aspects of project management.

There are different ***domains*** in project management. PMBOK 7 categorizes them as follows:

1. Stakeholders
2. Team
3. Development Approach and Life Cycle
4. Planning
5. Project Work
6. Measurement
7. Delivery
8. Uncertainty

As you see, each of them is a group of related project management activities with the same theme: They are areas of focus in the project.

Your methodology probably covers all of these domains, but the way it does may not be explicit, and it may only cover the minimum you need in every project. You can go through the domains and identify how each of them is covered in your methodology, and then, based on the nature of your project, decide whether you need to enrich that domain of your methodology to achieve better results.

Be careful with the following when making changes to your methodology:

- Don't make it too complicated, and don't add extra elements unless you have good reason to.
- Make sure your system doesn't lose its internal consistency; a single element may seem interesting on its own, but may not be compatible with the rest of the system.
- Integrate everything. Most elements are not effective unless they are properly

connected to everything else and work together.

Let’s review the domains and see what we can do in each of them.

4.1. Stakeholders

As a reminder, a stakeholder is a person or group with an interest in the project and who can impact it; for example, customers, suppliers, end users, team members, competitors, regulators, etc.

Stakeholder management is essential to the success of a project, and we explored different aspects of it in the principles chapter. All methodologies integrate stakeholder management into their system, and it’s usually done in a way that satisfies the average requirements of their target audience.

In the stakeholder management domain, the old-fashioned methods were mainly focused on external stakeholders. Then the Agile methods took the opposite approach and focused mainly on internal stakeholders. Modern approaches try to pay attention to both aspects.

If you need to be serious about stakeholder management, you need to have these elements in place:

1. Identify.
2. Understand and analyze.
3. Prioritize.
4. Engage.
5. Monitor.

You should repeat these steps continuously throughout the project and embed them into your existing method if they are not already there.

4.1.1. Identify stakeholders

Identifying stakeholders is not always easy; do you remember the example of building a shelter in the mountain?

It’s absolutely necessary to spend enough time at the beginning and identify at least the key stakeholders. That’s essential, because at the beginning we’re collecting information and usually want to decide whether or not it’s a good idea to run the project. Some stakeholders, such as regulators, may add serious requirements to your project that significantly impact its targets, such as cost, and you may end up rejecting the project because of that information. If you don’t go though that, you may start the project and spend money on it, and then realize the requirements of that forgotten stakeholder, and you’ll be in trouble.

Besides the upfront identification, you need to repeat it frequently, and always look for new ones, because

1. you may have missed some of the stakeholders, or
2. changes in the environment may have added new stakeholders.

As with many other aspects of project management, a good way of identifying stakeholders is to check the list of stakeholders from similar projects the organization has completed in the past. It may remind you of some stakeholders that you might otherwise forget about. This is why we must document the project information properly and archive it for future projects.

So, review your methodology and see where it requires you to identify stakeholders. If it’s not enough for your project, add more activities to your methodology to cover it, or merge it into existing activities of the methodology.

4.1.2. Understand and analyze

Immediately after identifying a new stakeholder, you should start thinking about it and answer basic questions such as the following:

- How can they impact our project? How much power do they have in doing so?
- What do they feel about the project? Do they have an overall positive or negative feeling toward you?
- What do they expect from the project? What makes them happy about you and what makes them mad?
- Etc.

When you wonder about those questions, you will also come up with a plan for how to engage those stakeholders and gain their support. When doing so, remember that your plan must be justifiable: The effort you put into engaging the stakeholder should be less than what you gain from their support. To be more precise, this also has to be higher than your ***opportunity cost***; i.e., more valuable than other things you can do instead.

The only thing that can override the justification is the ethical aspects. For example, you may be going to renovate a house and realize that the only neighbour is an old lady who may not know that she can contact the city hall and complain about inappropriate noise. So, the impact is low, but does it mean that you should feel free to make noise day and night? Of course not.

After extracting information about stakeholders and planning how to work with them, you need to document the information for future use. Your methodology may have a specific artifact for it with a name such as ***stakeholder register***. Otherwise, it may be part of a more general artifact in your methodology, or simply left to you to decide. If you need to have a stakeholder management system that is more serious than the average, you probably need to separate the list and create a register for your stakeholders.

When planning how to engage stakeholders, think about these two points:

- **Flow of information from you to them:** We usually have to inform them of the project and keep them up to date. This is necessary because they may have concerns for certain aspects that won't come up until they realize such aspects exist in the project. As usual, the sooner we find out about those things, the easier and less expensive it is to do something about them. Withdrawing information may seem like a solution to some people, but it's too risky, as well as being unethical, and for this reason should be avoided.
- **Flow of information from them to you:** You need to understand what they expect from the project. When possible, ask them for their opinion instead of waiting for them to come to you. Use creative ways that makes this communication more effective.

The form of communication should be selected based on the individual situation:

- Depending on the type of stakeholder, it needs to be formal or informal.

- It can be pull or push communication; e.g., an online dashboard that people can visit any time they want is “pull” communication, whereas a report you send to them is “push” information. Pull communications may require less work, but you should make sure that they remain effective and people don’t forget about them.
- There can be different mediums for communication; emails, printed reports, face-to-face meetings, etc. Some forms require more effort but also have the potential to be more effective in certain situations.

4.1.3. Prioritize

If you have too many stakeholders, it would help to prioritize them based on how justifiable your investment in that stakeholder would be, how powerful the stakeholder is, etc.; for example, you can sort your list of stakeholders so that the higher-priority ones are at the top.

This prioritization helps you direct your efforts, and, for example, if it’s a difficult time in your project and you can’t care for the stakeholders as much as you expected, you might only do it for the highest-priority ones and accept the risk of leaving the rest on their own.

Don’t forget that priorities can change as you progress, though.

4.1.4. Engage

Having a plan is not enough – you need to execute it. Take a look at your methodology to see how it embeds stakeholder engagement (communications, for example). Is that enough for your plan or do you need to add extra activities to your method?

Besides the plans that describe your overall strategy in engaging stakeholders, there may be many ad hoc decisions involved. When making these, soft skills such as negotiation and conflict resolution will be very useful, and it’s a good idea to invest in improving those skills in yourself and other key team members.

Unethical behaviors such as bribery are common in some countries. Don’t forget that you must behave ethically even if no one else does, and unethical behavior

such as bribery is non-negotiable.

There are subtle actions that can act like bribery or be interpreted as such by the second or third parties, and as such should be avoided. This potential depends on the type of stakeholder as well; for example, it's less sensitive if you meet with a manager from a partner company than it is when you meet a representative of a regulator. As a result, you may feel comfortable to meet the former over lunch and pay for the lunch, whereas the same setup may be borderline problematic for the latter stakeholder.

Imagine an open-source foundation: They go to different conferences and talk about their in-progress projects all the time, and use any opportunity to present it to like-minded people. On the other hand, some organizations with proprietary products and services, such as Apple, are sensitive about the confidentiality of their projects. There may be various constraints regarding the way you can interact with various stakeholders in your organization or country, and as a project manager, you should understand and respect all of them insofar as they don't have any ethical issues.

4.1.5. Monitor

When the subject is human perception instead of objective reality, a fully ***predictive*** method can't work well, and you should be ***adaptive***. So, instead of creating a sophisticated plan for how to engage each stakeholder, create a simple one, implement it, see how it works (***monitor***), and then improve your plan using that information.

Besides trying to understand the effectiveness of our plans, we should also monitor the stakeholders, because their interest and power can change over time, and those changes may require replanning.

It's also a good idea to have frequent evaluations to measure the overall satisfaction of stakeholders. P3.express has a monthly activity to do this, and it may be a good idea to implement it in other methodologies as well.

4.2. Team

Team members are also stakeholders and are therefore the subject of the previous domain. However, there are extra concerns for team members above what is covered in the stakeholder domain, and that's the subject of this domain.

Remember that a project manager should pay attention to this domain for two different reasons:

- There's a set of social responsibilities and ethical behaviors expected from a project manager, regardless of their impact on the project.
- Having a happier, more effective team is key to project success.

In other words, many of the things you do for the team also help the project, but that's not the only reason you do them. Don't seek justification when creating a safe and healthy environment for people, but do it as your ethical responsibility. See the positive impact on the project as a bonus.

This domain is a relatively open-ended one that you can expand based on your concerns and knowledge. We're going to briefly talk about three main topics in this domain now: the structure of the team, team building, and leadership.

4.2.1. Structure of the team

Methodologies give you a default, basic organizational structure for the team, but you can adjust or detail them to match your environment. When doing so (e.g., by adding new roles) be careful not to disturb the internal consistency of the system.

4.2.1.1. Project manager in Agile projects

While most Agile methods have project managers or recognize their existence (e.g., DSDM® and XP), Scrum is against it. Some people think that they can improve their Scrum projects by adding a project manager role, whereas it's not compatible with the way Scrum is designed. On the other hand, some people think that having a project manager is against being Agile, which is completely wrong; it's only unacceptable in Scrum and its derivatives, which is one way of being Agile but not the only one.

4.2.1.2. Centralized vs. distributed project management

Note that the PMBOK Guide is not about ***project managers***, but about ***project management***. The way project management is done can take two different forms:

- **Centralized:** In this approach, coordination and other project management activities are done by a team dedicated to project management, and there's someone on top of this team who's usually called the project manager.
- **Decentralized:** In this approach, there's no one dedicated to project management activities, but instead the activities are distributed among everyone in the project.

Both forms are valid and can be fine as long as the rest of the system is properly designed to match it and the approach suits the size and nature of the project. In general, a decentralized management system can be used in small teams; for example, you can't have a team of 100 people and have them coordinate themselves without a centralized form of coordination.

On the other hand, not everyone likes to have managerial responsibilities, and it can become a distraction for technical people to expect them to contribute to the management of the project instead of being focused on their technical work and having other people dedicated to providing them with facilitation and coordination services.

Decentralized system are more popular in the Agile community. While some methods, such as DSDM®, have a centralized management, Scrum uses a decentralized one. Most of the newer Agile methods are derived from or are highly inspired by Scrum, and therefore try to be decentralized, or at least pretend to be so. This is mostly troubling for those who try to create an Agile method for larger projects with multiple teams.

You may or may not have a ***project manager*** role in the project, but you always have ***project management***, either done properly and with a structure, or done poorly on an ad hoc basis.

4.2.1.3. Sponsors

Some people consider the ***project manager*** to be on top of the project team structure, whereas most methods have good reasons to have another role above their project managers: a senior manager in the company (preferably an

executive) who can

- provide the project with money and other resources with their organizational power, and
- make high-level, sensitive decisions based on their familiarity with the strategies of the organization.

Both of these are not possible for most project managers because they usually don't have higher organizational powers. The person who takes the role above is called a ***sponsor*** in P3.express and many other resources, an ***executive*** in PRINCE2®, and a ***business sponsor*** in DSDM®.

4.2.1.4. Labels

Be mindful of the fact that labels can be used in different ways. For example, a person called a ***project manager*** in your organization may actually act as a ***sponsor***, and someone who's called a ***team leader*** may be the one we consider to be a ***project manager***. So, don't be fooled by the labels and always pay attention to their actual role in the project and their set of responsibilities and authorities instead.

4.2.2. Team development

You must always put effort into various forms of team building. Some methodologies have cleverly designed activities that work as team building in addition to serving a different, primary purpose, and those combinations usually work much better than corporate-style activities dedicated to team building.

So, check your methodology to see whether there are enough of these two-fold activities, and if not, see how you can add them to suit your project.

You can also create a stronger sense of belonging by emphasising the mission of the project and the fact that everyone in the project has the same goal. Involving team members in helping with the improvement plans can contribute to this setup as well. Your work will be easier if your project has main value drivers other than pure financial gain; e.g., think about the researchers who were trying to build COVID-19 vaccines – as a project manager in such a project, you should

continuously remind them that they are all working together to save lives!

4.2.3. Leadership

As discussed before, you have to consider two main subjects in leadership:

- How can I empower the hidden leaders in the project?
- How can I improve my own leadership skills?

Create a list of leadership skills (motivation, problem solving, conflict resolution, negotiation, facilitation, etc.) and plan how you want to improve yourself in each of those using the existing resources and practice in the real world. It's also a good idea to expand this training to team leaders and other people who contribute to, or have the potential to contribute to the management of the project.

For example, let's say there's a young engineer in the project who seems to be interested in becoming a project manager in the future. As a project manager, you're responsible for providing them with the opportunity. So, if you're not great at facilitation and there's not been enough time to learn and practice it yet, maybe you could ask that engineer if they'd like to become responsible for that. Then send them to training courses to grow as a facilitator, and whenever there's a workshop, ask them to play the role. They would be happy to grow on the path they like and play an important role in the project, and you'd be happy to have someone to help you in an area you're not great in. A true win-win situation!

Speaking of win-win situations, it may be helpful to point out a general rule in negotiation and problem solving: Some people think they are great when they ***win*** everything in a negotiation. That may be a short-term win, but it would cause problems in the long term, and that's why the rule is to seek win-win situations in all negotiations and problem-solving attempts, as it's the only sustainable way. Would you like to learn more about this topic? Then it's a good idea to pick a good resource about negotiations and benefit from this great skill.

4.3. Development approach and life cycle

This domain is mainly about two related but different topics of ***development approach*** and ***life cycle***. These topics can become too complicated and lengthy,

and they are usually not required for an intermediate understanding of project management. Therefore, let's just have a brief look at them without going into too much detail.

4.3.1. Development approach

You can have a ***predictive*** or an ***adaptive*** development approach. We talked about their definitions previously, and if you don't remember it, I highly recommend rereading that section because every project manager must be aware of these.

4.3.1.1. Where "Agile" comes from

Adaptive methods have always been used in human history alongside predictive ones. When the first generation of commercially available computers was built, their limitations dictated a predictive method of development. While it was suitable to the early software development projects, as we progressed, their new capabilities changed the environment, and the predictive methods were no longer useful to the new IT development projects, although they were still forcing everyone to use them based on their past experiences.

Forcing of predictive approaches continued until their unsuitability in the new environment of IT projects became obvious, and some people started building new systems that were adaptive instead of predictive, and called them ***Agile***. Unfortunately, these circumstances created a negative feeling in many of them toward the predictive approach, which remains in the community to this day.

The Agile methods started from a blank slate, which had both negative and positive impacts. Those systems proved that simple systems can work as well as, or even better than, complicated ones. Many of them embedded human aspects into their methods, in contrast to other systems at the time that only used to stress the important of human aspects without embedding them into the system. This was invaluable, and will hopefully become the norm in all future systems.

On the other hand, the negative feelings, along with a few people's urge to create empires for themselves, turned Agility into a cult or religion for some, and closed their minds to everything else. Instead of learning from what other systems have

experienced and refined throughout the years, they've limited themselves to trial and error within a limited range of options.

In reality, both approaches are valid, and the choice depends on the type of product. People who think that any project can be Agile either don't understand what Agility is, or don't understand the diverse range of projects in the world. People who are against all forms of Agility are also mistaken and are losing out on a great opportunity.

4.3.1.2. What about "hybrid" approaches?

Unfortunately, following the common mistakes in the industry, PMBOK 7 talks about ***hybrid*** approaches as well. In reality, there are no hybrid methods for projects, because you can only be adaptive in a project when all its elements are adaptive, and if you make a subset predictive, it would block adaptation in the whole project. Yes, I tried, but I couldn't convince the rest of the team to remove it from the guide!

People who talk about hybrid approaches usually don't pay enough attention to the dynamics behind the scenes and focus rather on the superficial aspects of the methods, then as soon as they see a combination of the common elements of both methods in one system, they call it hybrid. This is another form of the ***Cargo Cult*** effect.

The only acceptable scenario in mixing approaches is to divide the product into multiple parts that are more or less independent of each other, and use a different approach for each. This setup is, in fact, a program with multiple projects, each using a single approach, and there's no real ***mixing*** of approaches in a single place.

4.3.1.3. Selecting the development approach

The right development approach for your project depends entirely on the product the project is going to create. The preferences of the team, project manager, organization, and customer have nothing to do with the choice of development approach.

So, do you have to adapt to the environment to understand what the final output should be, or can you predict it? More importantly, ***can*** you use an adaptive development approach at all?

In adaptive projects, we must focus on subsets of the product and specify, design, and build them without the rest. It must be done without the rest because if you design everything, then your choices for the future will be limited and you won't be able to adapt. Finally, the output has to be in ***increments*** that people can actually use, because that's the only way we can collect reliable feedback.

Now, think of a construction project. Can you specify, design, and build subsets of it without the rest? No. For example, you can't design a foundation without designing all the other elements, because their design determines their load, which is your input for designing the foundation. Also, what do you want to discover about a building that you can't predict?

On the other hand, if you're going to build a new piece of software for the public and rely on prediction, your chances of success would be extremely low, because we simply can't know how people will react to various products without carefully testing them in an adaptive environment.

Not every "IT project" needs to be adaptive. For example, if you're going to upgrade a data center, there's no human perception involved and it's best to have a predictive project. Also, if you're going to develop a software application for hospital equipment, aeroplanes, or the next rover that will be sent to Mars, there's no room for the trial and error that forms the backbone of adaptive methods, and so you must work with a completely predictive method.

Remember that what we do in adaptive projects is use the feedback to discover what the best path is for the next iteration of the project. Using feedback to adjust what was done before is not adaptive development. Moreover, the adaptation we talk about here is within the boundaries of a single project/product. If you run one project, learn from it, and use that to run the next project better, that would be a form of adaptation, but not the adaptive development approach for your second project.

4.3.2. Project life cycle

The choice of development approach has some impact on the project management methodology. Some methods, such as P3.express, are designed to be compatible with both approaches, whereas some, such as DSDM®, are designed to support only the adaptive approach. So, when selecting your methodology, you should be mindful of the development approach.

One important aspect is that the project management life cycle should be compatible with the development approach; for example,

- a cyclic management life cycle can be used for the cyclic development of adaptive projects as well as the more-or-less linear development of the predictive methods, whereas
- a linear management life cycle can only be used for a linear development approach, such as those in predictive projects.

Matching the life cycle with the development approach is a complicated topic, and there's no reason to worry about it as long as you pick a methodology that matches your development approach.

4.3.3. Product life cycle

Each project creates or modifies a product, and as a result, the project life cycle becomes a subset of the product life cycle. When working on the project, it's helpful to have the wider context in mind; for example, maybe you can spend €50k more on the project and add a feature that reduces operation costs by €2k per month. Is that a good idea?

Decisions related to the wider context cannot be answered within the boundaries of the project because they require a holistic view and access to information that may only be available to the highest levels of the (customer) organization. The project management system still has to identify potential ideas that may work in the wider context and report them to the higher levels for them to decide.

4.4. Planning

We were late for a meeting. We got into the car to go, and I started setting my GPS navigation and checking various routes. My companion told me, "Come on,

let’s go, we’re too late, we don’t have time for that!” But I was doing it exactly because we didn’t have enough time. A delay of less than a minute for planning could save us a lot of time later on by selecting the fastest route.

4.4.1. Planning for different development approaches

All projects need planning, with no exception. Some people think that Agile project don’t need planning, which is not correct. They just have a different type of planning:

- **Predictive** projects have either a detailed or a high-level plan upfront. When the upfront plan is high-level, there would be cyclic (e.g., monthly) stages for detailing them.
- **Adaptive** projects have a really high-level plan upfront. Some level of detail may be added on various cycles during the project, but each element is fully detailed only before it’s executed.

Anything we do must have a purpose. We don’t plan for the sake of planning, but we do so to

1. direct the project, and
2. enable monitoring and controlling.

The first of these depends on the development approach while the second depends on the project management methodology. That’s why planning is not an isolated concept and why we can’t judge how good a plan is without checking the development approach and the management methodology. Something that might be a good plan in one setup may not be so in another.

4.4.2. Continual planning

Planning is a ***continual*** activity, and regardless of the type of upfront planning, we **must** continually replan. Some people think they can spend a couple of weeks upfront creating a schedule, print it on a large piece of paper and put it on the wall, and then they are fine for the whole duration of the project. A plan that is created once and never revised cannot be of any use, though.

4.4.3. The subject of planning

When and how to plan is usually covered well in all methodologies, but the subject of planning is something you can spend more time on and tailor to your project when needed.

The old versions of the PMBOK Guide had a set of ***knowledge areas*** that were similar in some aspects to our current topic of performance domains, yet mainly different. They are, in my opinion, the best way of explaining various subject areas for planning as well as monitoring and controlling. Here are the old knowledge areas:

- Integration
- Scope
- Schedule
- Cost
- Quality
- Resources
- Communications
- Risk
- Procurement
- Stakeholders

This is only a way of structuring what you already know about various areas, but it's a well-formed list you can use to double-check and make sure everything is covered properly in your system. Your method probably covers all of them, but the level of detail may not be suitable for your project. So, the next time someone shows you a time schedule and says that it's their "plan", show them the list above and ask them where the other 12 areas are planned!

4.4.4. Types of schedule

Planning covers cost, quality, scope, risk, and many more areas. A core concept that should be planned for is time, which we usually refer to as a ***schedule***. Scheduling is assigning time or order to the elements.

There are two main forms of scheduling:

- Dependency-based
- Priority-based

In many projects, there are hard, unavoidable dependencies among elements; for example, you can't paint a wall before building it, and you can't build it before building the floor underneath it. Such projects are usually scheduled using a dependency-based method, where we create a network of those dependencies along with other determining factors and use that to calculate the orders and start dates. This calculation is done using either the ***critical path method (CPM)*** or a method similar to it.

Software applications such as Oracle's Primavera P6 and Microsoft Project are examples of tools built around the CPM methods.

On the other hand, some projects can be broken down into building elements that are not too dependent on each other. In this case, you don't need a dependency-based schedule, and a simpler priority-based method would suffice. In this method, you assign a priority to each element and then order them based on their priorities and importance, and then work through the list.

Adaptive projects ***require*** priority-based schedules, and if you believe that you can't create a true priority-based schedule, it may be a sign that the product can't be developed using an adaptive method. Predictive projects, on the other hand, can have either type of schedule.

4.4.5. Levels of planning

There are two major forms of planning:

- Planning the work
- Planning how to plan, monitor, and control the work

We can call them plans and ***meta-plans***. For example, your plan for risks is a list of risks you've identified and your responses to them. The whole plan can be a spreadsheet called the ***risk register***. Your meta-plan for risk would be a text document that describes the way you want to manage risks: the techniques you would use, the workflows, the artifacts, etc.

Each methodology tells you how to do each of these, and as such, some form of meta-plan is embedded into your methodology. However, it may not contain all the information you need in practice, and that's why some resources recommend creating these meta-plans to complement the methodology.

Usually, the systems that expect upfront tailoring (e.g., PRINCE2®) also expect meta-plans, while those that prefer gradual, ongoing tailoring (e.g., P3.express) don't enforce that. However, even in the second case, it would be a good idea to have them for areas that require more attention; for example, if procurement is a sensitive topic in your project, create a meta-plan for it. You can call it the ***procurement strategy***, the ***procurement management plan***, etc.

There are usually a lot of things in common among various projects run in an organization, and therefore, a meta-plan created in one can be usable (with some adjustments) in others. It's best to create the meta-plans in a central place and make it available to all projects. While the projects use the meta-plans, they would adjust them, and some of the general adjustments could then be fed back to the central place to be used in future projects.

What's described above is basically an important aspect of the first step of tailoring I explained in the principles section.

The centralized place that stores, distributes, and adjusts the meta-plan can have any name you want. A good option is a ***center of excellence***. Another common term is ***PMO***, which can stand for "project management office" among other things. Unfortunately, the term PMO is a vague one used for many purposes, and sometimes even has too much overlap with the project management system of individual projects, which is not a good idea.

4.5. Project Work

We first plan, and then execute. This is the case with all types of project:

- A **predictive** project may have most of the elements planned upfront and then executed, or it may have a relatively high-level plan and gradually detail it before starting each stage.
- An **adaptive** project doesn't have a big plan upfront, but there's a more

gradual form of planning done before each execution.

4.5.1. Plan-work balance

There's a lot to talk about when it comes to working, but we cover most of it in the planning and measuring domains. What's left to talk about working in a well-structured system is the relationship between working and plans:

- **Working without a plan:** Many projects don't have a structured system, and teams simply do whatever they deem necessary, guided by a vague mental image of the whole project. There may be a plan somewhere that's not used to direct what's best to do (maybe it doesn't even have the capability to do so), and the plan becomes more and more different from what's done in the project. Sometimes, there's a planner who struggles to update the plan, but even that's a one-way relationship where people do their work, and then the plan is adjusted to match what they have done and what they may do in the future.
- **Working without flexibility:** The opposite is to have a fixed, detailed plan and force everyone to act exactly as planned. No matter how well you've planned, this can never be realistic because projects are complex, or at the very least, complicated.
- **The balanced approach:** The practical approach is to have a two-way relationship between them, where we use the plans to see what we have to do, and reflect back the facts to adjust the plans and keep them realistic.

In P3.express, for example, there's a weekly cycle with a kick-off meeting. In this meeting, key team members or all team members (depending on the number of people) review the work planned for the upcoming week and check to make sure it's the best way forward and that there won't be any conflicts.

Whenever you adjust the plans based on realities, they change and give you new information. That information will then be used for measurements (discussed in the next domain) to see whether any ***corrective*** or ***preventive*** action is required.

So, there's a constant dance between plans and working. How is this reflected in your methodology? All projects are uncertain, but some are more so. In those cases, you may have to strengthen your methodology in this area.

4.5.2. Level of detail

There's an appropriate level of detail for plans in the management layer, and it's best to leave the rest to the teams, because otherwise your plans will become too complicated and inflexible.

PRINCE2®, for example, has a high-level plan created upfront, relatively detailed plans for each stage created right before starting that stage (the ***manage by stages*** concept), and finally, a fully detailed ***team plan*** that is up to the teams and not the project management layer.

Finally, while we insist a lot on planning, it doesn't mean that everything must be planned! The insignificant details are best left to ad hoc decisions of team members in order to keep the plans simple and allow more flexibility in the project.

4.5.3. Follow-up items

Being involved in doing the project work gives us a better chance of discovering ***follow-up items*** than abstract upfront planning does, and we shouldn't miss this opportunity. ***Follow-up item*** is a P3.express term that refers to any of the following:

- **Risks:** Have you discovered any uncertainties for the future that may impact the project? Most Agile projects have ***daily stand-up meetings***, where all team members get together for 15 to 20 minutes and each person answers three questions: What have I been doing since yesterday? What am I going to do today? What problems ***might*** I have? The last question is a simple way of identifying new risks in the near future. In projects that are not too small, you may not have the chance to have daily meetings like that, but you can still have a customized version of it that suits your projects.
- **Issues:** How about unplanned things that actually happened? You need to do something about them.
- **Improvement plans:** It's a great idea to have frequent activities for evaluating stakeholder satisfaction and using that, alongside other inputted data to see how you can work better in the future.
- **Change requests:** Does the customer or another stakeholder have a change

request? If so, first, we should assess the impact of the change on time, cost, quality, and other targets, and then prepare one or more proposals for how we can add it to the project. The requester can review the proposals and select one. Change is not a bad thing; it's only bad when you don't manage it in an integrated way.
- **Lessons learned:** Everything we do in the project can be used later as a lesson, with various levels of usefulness. It's great to keep this information and make it available to future projects.

It doesn't matter how simple and small your project is, it's always best to document all these items as soon as they are discovered, because otherwise they will consume more mental energy and there will also be the risk of forgetting them. After documenting them, you must make sure they are followed up on until they are closed.

P3.express uses a single artifact called the ***follow-up register*** to store every type of follow-up item, and it also has the rule that any follow-up item as well as every deliverable should have a ***custodian*** to continuously check it, make sure the related plans are being implemented, and report on it.

So, you should make sure your methodology covers all of these items, either with a minimalist approach like that of P3.express or in a more sophisticated form with multiple artifacts and processes, such as PRINCE2®'s.

4.6. Measurement

Plans have two main purposes:

- We use them to direct the project work
- We use them to evaluate the project work

The work follows the plan, and yet it never exactly matches the plans. So, we revise the plans based on facts to get realistic outputs for the remainder of the project. When we do so, we also get the revised targets of the project; e.g., your project was initially planned to be done in 16 months, but based on the current information, your estimate is that it will likely be finished in 18 months. In that case, what would you do?

4.6.1. Recovering from deviations

Most methodologies have a decision-making process based on the impact of the deviations. If the impact of the deviations is lower than a certain level, the team members decide how to recover from them; if the impact is medium, the project manager decides, and if it's higher, it will be escalated to higher levels to make the decision and tell the project team what to do.

Some people mistake the thresholds with whether or not a decision is required; for example, if a deviation is below a certain threshold, we don't need to recover from it. That's not what most methodologies such as PRINCE2® have in mind, though – The thresholds only determine who should decide about recovery, but recovery is always necessary because if we don't do anything about deviations, they pile up and become harder to fix.

4.6.2. Correction vs. prevention

When there's a deviation, you probably need to have a ***corrective action***. However, you can also find the root cause of the deviation and have a ***preventive action*** to avoid the same problem in the future. Unfortunately, most people only correct the deviations and don't put enough effort into preventing them.

If you must sacrifice one, sacrifice the corrective action rather than the preventive action; otherwise, you will be spending the rest of the project ***fire-fighting*** without enough time to do anything fundamental.

If you agree with this recommendation, maybe you can check to see whether you need to make it explicit in your methodology.

4.6.3. Targets

A common control is for the justification of the project; i.e., can we still achieve the project goals as expected?

So, your method probably has a frequent check for justification. If it doesn't, you probably have to add it; after all, you don't want to continue spending resources

on a project that doesn't serve you well.

What are the project targets?

The classical targets measured in projects are scope, time, cost, and quality. Some resources may add other targets such as overall risk and benefits as well. Justification of the project is usually a combination of multiple, or even all, targets.

How sensitive are these targets in your project? If the controls provided in your methodology don't match the level of sensitivity in your project, you need to adjust them.

4.6.4. Proper measurements

How do you measure performance?

Many IT development projects used to pay too much attention to the lines of code programmers wrote. There's the famous story of a company that hired a great programmer, and they asked him to report how many lines of code he had written after the first month, and the answer was a negative number! The programmer had cleaned up and fixed a lot of code, which mainly involved removing unnecessary code.

Most IT development projects are not like that anymore, mainly thanks to the Agile methods that have been fighting against these poor practices. However, some of the good practices of Agile methods have been deformed and used inappropriately as well. For example, most Agile methods work in fixed-duration iterations. At the beginning of each iteration, they pick a number of items from a big to-do list and create a simple plan for their iteration (which helps increase focus). Unfortunately, the main measurement for some projects has become counting the number of items the team has finished compared to what they had picked. This is obviously wrong, because it pushes them to be more conservative when planning their iteration, and then they end up delivering less because "work expands to fill the time available for its completion" (***Parkinson's law***).

Don't forget that what you measure in your project becomes the goal for people working in the project. Therefore, your measurement must be aligned with the real goals.

Measuring progress toward the abstract goals of a project is not easy; for example, measuring how much value has been created based on the organizational strategies. In view of this, we have to select the best possible proxies, which are the project targets (time, cost, quality, scope, etc.). More sophisticated alignment evaluations can be done in the portfolio management system using the target evaluations done in projects.

There are different ways of measuring the targets, but there's only one effective way: forecasting the targets for the completion of the project; for example, how much money do we need to finish the project.

4.6.5. Reporting

When you're done with measuring progress, you have to report it to some of the stakeholders. The team members should be aware of the measurement to align their work to it. Some management levels in your organization may require the information, and there may be an external customer as well, all expecting you to report progress to them.

In most cases, a single report doesn't suit every audience. So it's a good idea to create different types of reports. After reporting, check to see how it works and whether or not the main messages are properly understood. If not, you'll need to adjust your reports.

People are busy and easily bored, and therefore, simple, short reports usually work best. Some of your stakeholders may force you to send them long, detailed reports because it gives them a sense of comfort. Do this if you have to, but send them a short, focused version along with it.

Not all reports have to be on paper. Sometimes, you may realize it's best to add the information on a physical board somewhere people can see (sometimes called an ***information radiator***). If the target audience is not co-located, you can have an online dashboard instead. However, don't think that because you have a dashboard people will go and check it – maybe you'll have to email a brief version or a notification as well.

4.7. Delivery

There's the story that a manufacturer of elevators had complaints that their elevators were too slow. They ran multiple projects to improve their speed, with different levels of effectiveness, but the complaints remained. At one point, they wanted to initiate a new project to increase speed, and someone responsible in that project asked a fundamental question:

A: Why?
B: Why what?
A: Why do you want to make it faster?
B: Because the users and customers want it to be faster.
A: Why do they want it to be faster?
B: I don't know… I suppose faster is better!
A: Why is it better?
B: Well… it's a small box and people don't have anything to do there, so they just get bored.
A: So, the goal is to prevent boredom rather than to increase speed!

So they added a mirror to the elevator! It kept people busy looking at themselves, and the interesting point is that they even thought the elevators had become faster, although it was at the same speed as before.

4.7.1. Start with requirements

It's always risky to rush into the first "obvious" solution without staying with the problem long enough. That's why we have a certain process:

Requirements → Deliverables → Activities

Many plans start with activities or deliverables. Ideally, though, we want to start with requirements, then use the requirements to design the deliverables, and finally, see which activities are needed for building those deliverables.

This process is the same for predictive and adaptive projects. The difference is that most requirements are identified at the beginning of a predictive project, but must be identified gradually during an adaptive project.

So, what would you do if someone told you, in the middle of an IT development project, that they need to have a feature for the support staff to be able to log in as different users without having the passwords of those users?

What they've described is a deliverable, and you must ask questions to understand what the requirement is behind this deliverable. When you find the requirement, check to see whether there are other solutions for it, and which one would work best.

The traditional approach in adaptive projects, mainly created in XP (eXtreme Programming), is to describe what they want as a ***user story***, or ***story*** for short. There are different ways of forming a story, and the modern way goes something like this:

> As a member of support staff, I want to be able to log in as any user without having their password so that I can check what they see from their side and try to solve their problems (e.g., missing an order) instead of asking them lots of questions or expecting them to send screenshots.

This format contains a reason at the end that works like a requirement and encourages everyone to think about it. Besides that, it helps reinterpret the request; for instance, in the example above, we can see that they wouldn't need to log in as high-privilege admin users (that would be a bad idea) and it's best to make that clear right away.

Is this concern explicit enough in your methodology? If not, you may need to do something about it appropriate to the size and complexity of the project.

4.7.2. Deliverable breakdowns

Another topic related to deliverables is that most projects have too many deliverables, and it would be overwhelming if you tried to manage them in a flat list. For this reason, most methodologies use hierarchical breakdowns for deliverables, where the project is broken down into a few major elements, and those into their building elements and so on for a few more levels until you reach the level of detail required for the project. Many resources call it a ***work breakdown structure*** (WBS), but other names are common as well; for example,

PRINCE2® calls it a ***product breakdown structure*** (PBS) and P3.express calls it a ***deliverables map***.

Small projects can manage their deliverables with a flat list; for instance, micro.P3.express, which is designed for micro projects, doesn't have a mandatory ***deliverables map***. However, larger projects can benefit a lot from having a hierarchical breakdown of deliverables. If you have such a project and your methodology doesn't have one (e.g., most Agile methods), you may need to add one.

4.7.3. When to deliver

Sooner or later, the final output of the project, including all its ***deliverables*** will be delivered to the customer and the end users. The when and how of this depends on many factors:

- **Predictive projects:** Because of the more-or-less linear development in predictive projects, their outputs are usually not usable until the end of the project. (Think of a project for building a bridge.) Therefore, the output is usually delivered in one go, at the end of the project. On the other hand, some predictive projects may have major phases that end at different times and create usable subsets of the final output, and therefore, it may be possible to deliver them in a few steps instead of all together at the end.
- **Adaptive projects:** Adaptive projects create usable ***increments*** of the final output throughout the project to collect feedback and guide the way forward. These increments can be released to production and used by real end users, but when it's not possible (because of the operations difficulties), they will be given to subsets of end users, or even representatives of them. In this way, their delivery ***can*** be continuous.

An ***increment*** of the output is a set of ***deliverables*** that are ***usable*** by the end users; i.e., every increment is a set of deliverables, but not every set of deliverables is an increment. Projects can't be adaptive unless they have ***incremental delivery***, because useful feedback can only be generated by something usable, and not with other sets of deliverables that may be too abstract for typical users to understand.

4.7.4. Delivery risk

Having only one delivery at the end of the project is risky in all situations, because if something is not the way the customer and end users expected, it will be costly to fix the problem. Therefore, even in predictive projects where we may not have true deliveries throughout the project, we still prefer to review deliverables with the customer and end users and receive their formal or informal approval. This is usually embedded in methodologies, but if not, and you can predict problems for the delivery at the end of the project, it's best to adjust your methodology.

4.7.5. Limiting work in progress

Limiting work in progress is an essential concept in ***lean*** manufacturing, which has had a great impact on Agile methods. However, it's not limited to adaptive projects, and every project can benefit from it.

We don't have to avoid all forms of parallel work, but there's an optimal level of parallelism for each type of work that is usually much lower than what most people expect. When you spread your capacity into too many deliverables at the same time, it becomes too distracting, and everyone would be tempted to start working on a new deliverable as soon as one runs into difficulties. Solving issues and finishing something immediately is usually easier than leaving it for the future. Remember: It's best to focus on finishing everything sooner rather than starting it sooner.

It's helpful to have an element in your methodology that encourages everyone to finish and close deliverables before moving on to new ones. On the other hand, the same element can be connected to an informal (or formal, if needed) acceptance by the project manager and then by the customer, which in turn helps with the previous point about avoiding issues at the end of the project.

4.7.6. Quality of work

So, it's a good idea to have a process for the project manager to check completed deliverables and give their informal approval. However, how can a project manager do so when they are not a technical expert?

This control is focused on making sure the deliverable satisfies its requirements and expectations. That can be translated into scope and quality. Before starting the work, you get help from technical experts to define the scope and quality of the deliverable and make sure they are aligned with the requirements and expectations. When the work is finished, you then ***help*** those technical people by reviewing their work based on the definitions to make sure nothing has been missed.

To be able to do so, you need to define scope and quality upfront, or at least just before work on each deliverable starts. For example, PRINCE2® has an artifact called ***product description*** for this purpose, and P3.express uses comments in its ***deliverables map*** to store the information. Check your methodology to see how it's done and whether or not it's appropriate for your project; if not, tailor it by increasing or decreasing the level of detail and formality, turn it into a separate artifact, or merge it into an existing one, etc.

Agile projects usually involve writing down the items on sticky notes or index cards. One side can contain the user story while the other side contains the extra information about acceptance criteria. On the other hand, many of the items in IT development projects have similar acceptance criteria, and therefore, instead of repeating them, we can extract those common criteria and store them in a single element. That element is called a ***definition of done*** in many Agile systems.

4.8. Uncertainty

There are various topics in uncertainty, but the most important one is risk management. As we talked about before, you have a proper risk management system as long as you have a structured system for identifying, analyzing, and planning responses to risks, and then you make sure the responses are acted upon and re-analyzed.

4.8.1. Level of sophistication

How you can implement a risk management system can vary widely. For example, the minimalist approach of P3.express has only one artifact called the ***follow-up register***, used for storing information about risks as well as issues,

change requests, improvement plans, and lessons learned. They are in the same place for two reasons:

1. You can use a simple process that applies to all of them without making your project management system too complicated.
2. Those elements morph from one type to another as time passes; for example, an uncertain event in the future, which is a risk, can actually occur, which then makes it an issue. When you work on it and close it, it becomes a lesson learned.

While this approach works well for most projects, a mega-project with a high degree of complexity may require a more detailed approach. In those cases, you can separate the element into different artifacts and use different processes for them.

PRINCE2® and many other resources use a separate ***risk register*** for storing the risk data. This register, which can be a spreadsheet, contains the analysis information as well as planned responses.

A single risk can have multiple responses, and a single response can serve multiple risks. Therefore, if you still need to make your system more advanced, it would be a good idea to separate risks and responses into two different tables and then create many-to-many relationships between them.

If you need to be even more advanced after separating out risks and their responses, you can even break the risk table into causes and effects, because a single cause can have multiple effects/risks, and each risk can have multiple causes. You can even go one step further and convert it from two tables with many-to-many relationships into a graph.

In practice, there’s no end to how complicated you can make this system, and you should always build one that is appropriate to your project. If you have doubts about the appropriate level of sophistication, though, err on the side of the simpler one.

4.8.2. Quantitative analysis

Your response plans to risks must be justifiable; for example, paying €1,000 per

month in insurance to avoid a low-probability event that may cost millions of euro may be justifiable, but it won't be justifiable if the damage is only a couple of thousand euro.

Many methods leave it to you to judge the justification of responses in an intuitive or ad hoc way, and that's fine in almost all projects. However, some sensitive projects require a more structured way of analyzing risks.

In that case, you can extract the required data and use the ***Monte Carlo*** analysis to combine them in various forms and give you the probabilistic outputs before and after applying the responses. For example, you may see that there's an 85% probability of finishing your project in 22 months, whereas, if you apply the responses, which cost you €80K, you can increase that probability to 98%. Then you can check whether or not it's worth the investment.

This type of ***quantitative analysis*** consumes a lot of resources and you have to make sure you really need it before adding it to your system.

4.8.3. Relationships with higher management levels

Everything we do in a project should be aligned with higher levels of management, but there are different degrees of importance, and risk management is one that requires a lot of attention in this area.

The main reason for the sensitivity of risk management is that some risks impact more than one project in a company, and when they do, it's best to respond to those risks in a holistic, unified way. To this end, you should ensure that the ***portfolio management*** layer of the organization maintains a list of overall risks that impact more than one project, along with their associated risk responses, and makes it available to all projects. Conversely, the list of risks from each project should be available to the portfolio management level so they can frequently review and identify risks that could impact multiple projects, and move them from the project layer to the portfolio layer.

You may be wondering what to do if your organization doesn't have a portfolio management system. Unfortunately, there are countless problems that can arise when there's no portfolio management layer, and there's no solution for that other than implementing proper portfolio management! However, if you're a project

manager in an organization that doesn't want to have a structured portfolio management system, maybe you can talk to other project managers and convince them to have monthly meetings to get together, share lessons, and talk about high-level risks across projects.

Remember that project management is about doing things right, whereas portfolio management is about doing the right things.

5. Where to Go from Here

I hope you've enjoyed this brief overview of the PMBOK Guide, and especially the fact that it was about the real essence of the PMBOK Guide rather than a one-to-one summary of the content of the official manual. If you check the manual, you may be surprised to see topics that we've not explicitly explored in this book, but don't worry – their important elements are covered implicitly throughout the book.

Many people try to create a project management system entirely based on the PMBOK Guide, and they fail simply because that's not the purpose of the Guide – it's not a methodology. That's why my focus was on explaining what it really is, and how you can use it in your projects.

There are two ways you can get help from the PMBOK Guide in your projects:

- The first is to use the principles to understand, re-interpret, and evaluate everything in your projects.
- The second is to use its ***performance domains*** to tailor and enrich your methodology.

That brings us to the topic of methodology, and that's why we had a separate chapter to introduce a few of them.

If you're interested in learning more, here are a few ideas:

- **Structured guides**
  - **PMBOK Guide:** You can buy a copy of the official manual from your favorite bookstore. Alternatively, if you become a member of PMI, you can download all the publications for free. If you're interested in learning about the old PMBOK Guide, you can read the new publication called ***Process Groups: A Practice Guide***.
  - **Open PM²:** Open PM² is open and available for free from the European Commission's website. It calls itself a methodology, but it's an unusual use of the word "methodology", and in my opinion, it's a guide like the PMBOK Guide rather than a methodology (one that shows you the path).
  - **APMBoK:** This is mainly known in the UK. If you like to have a wide perspective, it would be a good choice. I like it because it structures and

covers a few concerns that are overlooked in many other resources.

- **Methodologies**
  - **PRINCE2®:** This is one of the best general purpose project management methods. Like other reference material, many people find the official manual complicated and dry. If it's the same for you, you can read one of the many books written about it instead. However, when doing so, make sure you select a book that is focused on understanding PRINCE2 rather than one that is just focused on passing the PRINCE2 exams.
  - **P3.express:** This is an open, minimalist project management methodology, relatively newer than the alternatives. Its manual and a simulated project for practice are available for free on https://p3.express.
  - **micro.P3.express:** P3.express is designed for small, medium, and large projects and doesn't target micro- and mega-projects. micro.P3.express is a variation of it designed for micro-projects with a few team members. Similar to P3.express, it's open, free, and minimalist.
  - **DSDM®:** This is a sophisticated, well-structured, first-generation Agile methodology. You can find a free online version of its manual on their website.
  - **Scrum:** There's an (official) ***Scrum Guide*** available online, but it's a short document and you may prefer to read a book about it instead. There are so many books about it, but many of them are only collections of clichés about Scrum, so make sure you select a deep, meaningful book about it.
- **Specialized guides**
  - **Behavioral competences:** There are many great books about critical thinking, leadership, motivation, negotiation, conflict resolution, etc. It's a great investment of time for a project manager to learn more about these, and there are great books on each topic that you can enjoy.
  - **Technical competences:** Depending on the environment of your projects, you may want to use specialized books to learn more about scheduling, risk management, quality management, etc.

Finally, as a project manager, it's also a good idea to learn a little about portfolio management as well.

Live long and prosper!