NUPP

Nearly Universal Principles of Projects

This is a downloadable version of the online manual (https://nupp.guide), generated on 2024-07-19. Please check the website for newer versions.

NUPP is a collection of nearly universal principles of projects: those we'd do well to follow in all projects, regardless of the methodologies and approaches that we use, to maximize our success.

Each of the available resources and methods for running projects relies on some of these NUPs (nearly universal principles). However, the following points need to be borne in mind:

- It's usually not all of them, and it would be helpful for practitioners to consider all NUPs instead of a subset.
- The underlying principles are usually not made clear enough in resources and methods, and most practitioners are so engaged in practical details that they forget about principles and do things that are not compatible with them.

NUPP is compatible with all the major methods, systems, resources, and frameworks such as PRINCE2®, PMBOK® Guide, P3.express, PM², DSDM®, XP, and Scrum. It may not be compatible with certain interpretations of those systems, though, and that's where NUPP tries to encourage practitioners to reconsider their interpretations.

NUPP is a collection of the following NUPs:

- NUP1: prefer results and the truth to affiliations
- NUP2: preserve and optimize energy and resources
- NUP3: always be proactive
- NUP4: remember that a chain is only as strong as its weakest link
- NUP5: don't do anything without a clear purpose
- NUP6: use repeatable elements

Version 1.0, June 2019

NUP1

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.

Example: Agile vs waterfall

A group of highly enthusiastic people who were brave enough to try adaptive development approaches in IT development at the time when the norm was to use predictive approaches got together and called their approach "Agile". This was a great initiative to not limit choices to what seemed to be necessary. There are still many enthusiastic and result-oriented people in the Agile community, but unfortunately, there are also some people in this community who turn Agile into a cult and consider all outsiders as enemies. This causes problems in multiple ways, including the following:

- It doesn't let them learn from anyone outside their group
- It discourages outsiders from learning from them
- It makes belonging to the group more important than the real purpose, which in turn prevents many of its members from learning the real meaning of Agility

This problem can be significantly reduced, if not removed, by using "Agile" only as a label that refers to a development approach rather than as a community with members; and by having people who consider themselves

creators, problem solvers, and leaders, who see Agile simply as one of the enablers under their belt rather than as their identity.

There's no Agile-waterfall war for real professionals.

Example: PRINCE2® vs PMBOK® Guide

There are many professionals in the community who associate themselves with either PRINCE2® or PMBOK® Guide (usually because of their geographical location) and are not familiar with the other. We can all have preferences toward certain resources, but not as our identity, and more importantly, we must familiarize ourselves with all of them to widen our perspective and choices.

The real professional is open to all ideas, looking for them, learning about them, and using them as and when needed, without affiliations.

Example: Continuous learning

Affiliations satisfy the person due to the feeling of belonging they engender, but don't push them to learn, and sometimes even discourage them from learning for fear of losing them. When you're a free person, an expert without affiliations, you need to fill in the gap with learning: with continuous learning.

What we believe in today is not the truth. It's merely our best understanding so far, which has to be improved as we go on. There's something wrong if one's ideas are exactly the same as what they were a few years ago. This is even the case for NUPP: if you come back after a few years and see the exact same thing, you should become suspicious.

Example: Openness

When objecting to someone, make sure you're aiming your objection at the idea, and not the person. This helps prevent a lot of tensions. In a similar vein, when someone is objecting to or about you, try not to interpret it as a war against you, but rather a discussion of your ideas, and stay open to it. Don't

listen to respond, listen to understand; and work with the other person to improve the idea.

Some people may intentionally target you instead of the idea, in which case, you should help them focus on the idea instead of on you before proceeding, and try to keep it like that throughout the conversation.

NUP2

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.

Example: The 80/20 rule

A large portion of the possible benefits of project management can be gained by expending a small portion of the effort. In most cases, targeting 100% of the possible benefits is very expensive and unjustifiable. You need to consider this rule in everything you do and encourage others to do the same.

Example: Decision fatigue

We use a single source of mental energy for making all types of decisions, and also to express willpower. If you use up a lot of this source on making unnecessary or unimportant decisions, you'll have less energy for the important decisions, which may lead to poor results. This is one of the reasons you should avoid micro-management (the "manage by exception" principle of PRINCE2®).

Conflicts that are about ideas can be helpful, but those that are about people are usually harmful to the project, and one of the consequences is that it drains the mental energy of the team members. If you notice such a conflict, you should do your best to find the root cause and solve it.

Example: Take care of yourself!

The decisions you make and the willpower that you express use up your source of mental energy. On the other hand, this source is filled with energy when you sleep and eat properly. So, you should take good care of yourself: make sure you have enough sleep and rest, and eat well. If you have harmful habits or problems with sleep, you don't have to deal with it alone; there are many specialists who can help you fix such problems. Physical activity may also help with this source of energy, although studies are not yet conclusive on this matter.

Try to encourage the team members to do the same as you do. First, make sure they work at a sustainable pace and without too much overtime. Then, if you have the choice, try to offer energetic, healthy food, snacks, and drinks during work time.

Example: Work-life balance

Many of us love what we do, but that's still not everything we need to have in life. In the same way that you optimize your work, you should be planning and implementing ideas in your personal life, in ways that make it a joyful, happy one. When you're happier, you can be more successful at work too. If you can, try to encourage your team members to do the same.

Example: Motivation

Motivation is a complex concept. There are some interesting and useful resources on the topic, as well as many more unscientific ones. Nevertheless, you should learn about it and use it continuously, as it increases the mental energy of the team and the possibility of achieving better results for the project.

Motivation can be as simple as letting people know that you've recognized their good work by a kind smile or a simple "thank you". However, you need to be careful, because many of the common forms of motivation, such as small monetary rewards, have a negative effect.

Example: Collaboration and teamwork

People who are collaborating may sometimes have the power to create better results, but more importantly, humans are social and enjoy being part of a group. If you can remove the negative aspects of teamwork and create a pleasant environment, there will be happier team members in the project.

You should be careful, though, because people are different, and some need more relaxed, focused, and solitary time than others do; it's usually a balancing act.

Example: One-team culture

There's a tendency for different stakeholders to create or consider subgroups and cause tension with other groups; for example, people who are focused on the business aspects of the project vs. people who are building the product. This tension consumes a lot of energy from both sides, which is one of the reasons we should try to build a one-team culture, where everyone works together towards the same goal.

Example: The wisdom of crowds

When a large number of people with diversity get together and work in a facilitated environment, there's the potential to get very good results, ideas, and solutions, that may be even better than those coming from single experts.

If you have that option, you can use it regularly to ask team members to help you solve difficult problems in the project. Beside the possibility of getting good results, it also allows team members to know that their opinions are appreciated and that they play an important role in the project, which in turn increases their level of energy. Activity E02 of P3.express is an example of using the wisdom of crowds in projects.

Example: Chief project facilitator

If you are a project manager, most of the things you do have a facilitation

nature (or at least, should have). On the other hand, you may see that the team members have had bad experiences with project managers in the past, and that these experiences are impacting on their relationship with you: a portion of their energy is spent on analyzing your behavior for potential threats instead of trusting you. In that case, you can change your title from project manager into Chief Project Facilitator. After all, that’s what you really do in the project.

NUP3

Always be proactive

There's a natural tendency in us to be reactive. It can help us preserve our energy dealing unimportant matters, or it may give us better results when we are dealing with something in which we're completely incompetent. Those situations are different from our projects, and here we can get better results by being proactive.

Example: Planning

If you want to drive to a new location and you're late, you can start driving immediately to "save time", and deal with possible problems when they arise. The proactive approach is to take some time at the start and set your navigation system to give you the fastest route based on the traffic and possible accidents and blockages and then drive; that's spending time before executing, to avoid problems later on and thus save time in the end.

In contrast to what some people think of Agile projects, planning is always necessary, and it's only about the type and level of details in plans. Planning before executing is a proactive approach.

Remember the quote: give me six hours to chop down a tree and I will spend the first four sharpening the ax.

If it's a predictive project, you may spend 4 hours sharpening your ax, because you're sure that you're going to chop down a tree. In an Agile project, you're not sure if you're going to chop down a tree, gather broken branches, harvest turf, mine coal, or something else. Nevertheless, you still need to have a general preparation for all of those (know where the nearest hardware store is), and have a specific preparation (sharpening) when you're going to focus

on a certain solution; that's planning.

Example: Planning the planning

Planning the way we're going to execute the project is a proactive approach. This proactivity can even be extended by planning the way we're going to plan the execution; that's the management plan concept of the PMBOK® Guide, management strategies of PRINCE2®, and approaches in DSDM®.

Example: Continuous planning

Reality rarely matches what we have planned, and that's OK – but, we have to continuously adapt our plans to make sure they stay realistic and practical. We should do it as soon as adaptation is needed, and not when we run into problems. That's a proactive approach.

Example: Risk management

The whole concept of risk management is based on proactivity: when facing uncertain events, instead of waiting to see what happens and then reacting to them, we think about possibilities and impacts, consider responses, and probably do something about it before it happens.

Note that what we do in projects is serious; it's sometimes about people's lives.

Example: Define roles and responsibilities

You can leave the project team members to work without clear roles and responsibilities, and sooner or later a form of roles and responsibilities will emerge, but it's too expensive and may not work well after all. The proactive approach is to define them early and adapt them as needed. This makes working easier for everyone, and they can focus on producing something, instead of deciding who does what.

The number and variety of roles depend on the type and size of the project; it can be a simple definition such as the one in Scrum, something moderate such as that in P3.express, or something comprehensive like the ones in DSDM® and PRINCE2®. However, don't forget that the role descriptions in these methods are only about management activities, and that you always need to add role descriptions for technical aspects as well.

Example: Available choices

Should you close the project prematurely or continue with it?

There are rarely only two choices, even if the question implies that. You need to have a proactive approach and consider all your choices before making a decision. Maybe you can rescope the project; maybe you can pause it until something else becomes clear; or maybe you can change the project approach (e.g., outsourcing), etc.

Example: Critical thinking

We all have many biases that help us survive on one hand, and fool us into making bad decisions on the other. When it comes to making important decisions about the project, it's best to pause for a while and consider all biases that can impact our decision before they cause problems.

As a reference, you can use the list of cognitive biases given in Wikipedia: https://en.wikipedia.org/wiki/List_of_cognitive_biases

There are even decision-making frameworks that you can use to make better decisions. At first, it may be distracting and even annoying to use them, but soon you get used to them and gain advantage from them without much conscious effort.

Example: Transparency

We don't like being late in the project or having any other kind of problem, but it doesn't mean that we should hide it. You should be transparent and let the

stakeholders know, because some of them may be able to help you, and furthermore, they will know about the problems and their consequences sooner or later, and some of them may require early actions from their side (e.g., to accept the negative consequence).

Example: Communicate effectively

There may be many cases in which you send reports to stakeholders and they don't give you any feedback. You may believe that everything is fine just because there is no negative feedback, although it may not be so. You have to be proactive and check to see whether they've really used the report and whether it has served the purpose, and use the input to adjust your method of communication; otherwise, this hidden issue may cause serious problems later, when it's too difficult to fix.

Example: Take responsibility

It's easy to blame others for poor results. For example, you may want your organization to give you full authority to change everything in the project and do it perfectly, but they don't, and as a result, the project fails. This is not a proactive approach.

The proactive approach is to take responsibility and do everything you can within the constraints. You cannot expect the organization to fully trust you and give you everything in the hope of getting good results, especially when they have seen so many failed projects. What you have to do is to make one small improvement within the constraints that are set, use that to gain a little trust, a few more resources and a little more toleration for constraints, and then use that for a slightly bigger improvement, and carry on like that until you reach the optimum target.

NUP4

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.

Example: It's all about the deadline!

Let's say you're building something for the Olympic Games. It has a very serious deadline, which makes time management very important. Is that right, though? What if the quality is so low that it necessitates repeat work after a while. That would impact on time, So, that makes it time and quality. You may have a fancy garden listed in the original definition of the project, but you know that if there's not enough time, you can skip it and just cover it with grass, as long as you have considered this possibility in time and have prepared for it. So, scope management is also important. Now we have the scope, time, and quality domains at the center of our attention.

Have you heard of that famous example where president Kennedy meets a janitor in NASA and asks him what he does, and he replies, "I'm helping put a man on the moon"? Doesn't having that type of people in the project help meet the deadline?

As you go on, you notice that every single domain in the project contributes to time management, and you can't meet the deadline with an acceptable level of certainty unless you pay attention to all domains.

Example: Cherry picking

When people are faced with a variety of methods, sometimes they start cherry picking and create a mix of everything that seems interesting from different systems. This usually doesn't work, because elements do not work in isolation and have to be compatible with each other. Any additions or changes to a system should be made from a holistic viewpoint.

This is why we sometimes see contradictory elements in different methods; something works well in one system, and its opposite works well in another system. That element is not right or wrong on its own.

The safest approach is to select a methodology for the project, tailor it, and then cautiously add new elements to it by considering the consistency of the whole system.

Example: The anti-process approach

One of the best things Agile methods have done is to draw attention to human aspects. The Agile Manifesto gives more value to individuals and interactions compared to processes and tools, although this may not be a fair comparison. Almost all methods say that human aspects are important, and the real difference with Agile methods is that human aspects are an embedded part of their processes, rather than a simple suggestion. So, it's not about a competition between human aspects and processes, but rather about the way human aspects are seen in the system.

There's no doubt that some people try to replace the human aspects by having more sophisticated processes, but that's only a misuse. Even the opposite exists as well: people trying to replace processes with human aspects, which doesn't work well either.

Example: These are all the domains you need

When thinking about the domains, you should be careful not to miss any of them. What are they, though? If you check the fundamental resources, you will receive different answers; and yet, none of them is the whole truth.

PRINCE2® themes are domains, but those are only the domains that play a key role in the methodology. The other domains are only implied.

The PMBOK® Guide is not a methodology and can formulate it much better with the ten knowledge areas. However, these are interpretations of all domains based on the PMBOK® Guide's perspective on the project, rather than a neutral one (not that there necessarily is a neutral perspective). For example, human aspects don't receive a lot of attention in the PMBOK® Guide.

A good source of information about the domains is ICB. However, it's not about the domains, but about the competencies that are required in the project. It doesn't have a one-to-one relationship with the domains, but it does help a lot in identifying them.

There's no list of domains in NUPP, primarily because it's a meta-system rather than a system, and also because the categorization of the domains depends on the type of project and its environment; e.g., a routine construction project may need a different perspective from a creative research project.

NUP5

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.

Example: Portfolios and programs

If you’re involved in selecting and initiating projects, you should make sure that you focus on the benefits and the problems that are required to be solved rather than the product that is going to realize those things. The classic example is the manufacturer of elevators who used to receive complaints about the speed of their elevators, and so had ran multiple projects to increase speed, without complete success. That continued until they decided to focus on the problem (people’s boredom or discomfort) instead of the “natural” solution (faster elevators). The result was that they added mirrors to elevators, and it solved the problem simply and cheaply.

Remember that project management is mainly about doing things right, while portfolio management is about doing the right things. It doesn't matter how well you run your projects; it won't work well if you're doing the wrong projects. It's all about having a purpose.

Example: Project as a whole

The flexibility of the product varies between projects. In some IT development projects, the product is completely flexible, and its final form depends on the feedback that will be generated by the increments of the product during the project, which requires an adaptive (Agile) approach. This is in practical terms a combination of the portfolio, program, and project layers, and needs maximum attention paid to the ultimate goal. It is a good idea to document the purpose and have it accessible; it's one of the purposes of "product vision", as used in some Scrum projects. The attention to business value in Agile projects is their way of ensuring alignment with purpose.

In other types of project, where the product is relatively fixed and there are other mechanisms to ensure that the identified product will satisfy the purpose, it's possible for the project team members to shift a large part of their focus from the purpose to the product (hence the "focus on products" principle of PRINCE2®), but at least minimal attention to the purpose is still required to ensure that what is being built will satisfy the purpose, which can be done by comparing the forecasted benefits with the expected benefits (hence the "continued business justification" principle and the "business case" theme in PRINCE2®).

When a project is run for a external client, the client would have their own purpose for the project, and your company would have a different purpose. You should understand and use both of these to really succeed.

Example: Project monitoring

Focusing on the ultimate purpose is necessary, but it may be too abstract for many of the day to day uses. That's why a hierarchy of drivers is needed to make it more practical. First, the justification and benefits of the project are

defined based on the ultimate purpose, and then you will have explicit and implicit targets for project variables (e.g., time, cost, and quality) to satisfy the justification, which in turn will satisfy the ultimate purpose. These are lower-level purposes that are useful for many of our daily tasks.

When it comes to monitoring, the low-level monitoring of the project will be done using the lowest level of variables, because it may not be possible to track the ultimate purpose. In this case, you should still have the purposes in mind: what is the purpose of monitoring?

A normal and acceptable answer is to see whether we're on track, and if not, to trigger a certain reaction that will bring us back on track or adjust the targets and see whether it can still satisfy the ultimate purpose. Our measurements should therefore be able to help with this low-level purpose, and the appropriate measurements are forecasts for the variables at completion; e.g., when would we be able to finish the project? How much money would we need to finish the project?

All other measurements, such as the planned and actual values to date, are just interim values that you need in order to calculate the forecasts, not the final values that you use for managerial purposes.

Example: Documents

No matter what development approach you use in the project, planning is always necessary. The important question is the level of detail in each type of plan. If there's not enough detail in the plan, the plan won't be able to contribute enough, and executing the project will be based on ad hoc decisions rather than integrated, holistic ones. On the other hand, many people try to be cautious and add too much detail to their plan, which makes it too difficult to prepare and maintain, and too rigid for the uncertainties of the project. As a result, the plan becomes unrealistic and useless.

The best way to decide about the level of detail is to have a purpose in mind for every element in the plan. For example, if you're considering the addition of resources to the plan, you should have a purpose for it: How are you going to use it? How will it help the project? How much effort will it take? and is it worth it?

It’s sometimes about deciding whether you want to have a certain element in your plans, and sometimes about the way you want to plan or prepare something. Whether it be a business case, a project charter, or a report: you should still ask yourself why you’re preparing that document and how it can help the project.

Looking for a “template” is the opposite of doing something based on a purpose.

Example: Status reporting

It’s common to have really long status reports in many projects. Based on this NUP, we should ask ourselves what the purpose of the report is and how we can achieve that purpose regardless of what most other people may be doing.

This can, in many cases, lead us to prepare very simple one-page reports that stakeholders really read and understand instead of long reports. This is paying attention to purpose.

If you prepare one-page reports, though, some people may object about you and think that you don’t have a “proper” monitoring system in place. In practice this creates a second purpose for you (besides the first purpose, which is helping stakeholders understand the status of the project), and to satisfy that, you can simply create a second type of report that is very long. However, you won’t mix that with the first report, because these two purposes are not the same.

Example: Business case and project charter

Preparing documents such as a business case and a project charter is usually a boring, fruitless, bureaucratic necessity for most people, while these documents all have valid purposes that apply to most projects. If you try to find a “template” and fill it in, the work is just wasted; whereas you can instead check the real purpose of those documents, see whether and how they can be helpful in your project, and then prepare the document in any way you like, just to satisfy those purposes: that will be the right document.

While you're thinking about the way you can prepare the documents to satisfy their purposes, you may not think of every scenario and may miss something. To avoid that, you can check to see what resources such as PRINCE2®, PMBOK® Guide, P3.express, and DSDM® suggest, and then evaluate those suggestions based on the purposes.

Example: Post-project

While the project has a specific purpose, and we can consider that purpose throughout the project, the realization of the purpose is mainly evaluated based on forecasts made during the project. We should not, however, forget about it when the project is finished. It is important to check the realization of the goals after the project is finished, because

- we can see how the ultimate goals are achieved and learn from that for future projects, and
- sometimes the goals are achieved only when we carry out some additional tasks after the completion of the project, and understanding the necessity of those extra tasks needs monitoring.

Post-project monitoring is a necessary step in being purpose-driven. It's mainly the responsibility of program and portfolio management systems, and because most companies do not have such layers of management, this step is usually neglected. That's why methods such as P3.express and DSDM® add this step to their project management methodologies.

Example: Simplicity

The world is complex and chaotic, but our models are abstract approximations that reflect parts of the world, and hence, they can be simple. On the other hand, simple systems usually work better, because we can be focused on the main idea.

Many of us try to get better results by adding complexity to our systems, hoping that it will match the complexity of the world, even though in practice this makes the system difficult to work with and usually blocks us from

satisfying the purpose.

Example: Tailoring

A piece of software for streaming music has a very different condition from one that will be used for equipment in a hospital or a on plane where people’s lives may depend on it, or from a piece of software that will be used in a satellite where it’s supposed to work for many years without crashing, and they are all different from building a summer villa, a fire fighting station, and a process plant.

When you have the purposes in mind, you will better understand how to tailor the systems and artifacts for different projects.

NUP6

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.

Example: Quality checklists

A checklist is a simple example of a potentially repeatable element that many people use in their personal and professional lives. Take quality criteria of a deliverable, for instance:

- First, you may create a checklist of all criteria, which is a form of planning.
- What NUP6 recommends is to try to generalize it: are there other similar deliverables in the project? In that case, prepare a general quality checklist for that category of deliverables and use it for all of them. If there are some variations, keep the generic list, and add a few extra items for the individual deliverables. Now you have repeatable checklists.
- Once you have prepared generic checklists for various types of deliverables, you may find elements that repeat among them, which suggests a virtual parent category for them. In that case, instead of repeating the items for all those generic checklists, you can extract them and put them in a parent checklist. In the end, you will probably have a single generic checklist for the whole project. Scrum's "definition of done" is an example of using project-level checklists for quality (possible among other things). By doing this, each deliverable will belong to a hierarchy of categories and should satisfy the items that appear in checklists of all categories in their chain.

By doing this, an item in the parent checklist will become repeatable for all deliverables underneath it, which saves time and energy in planning and execution.

More importantly, once you do this for one project, you can tailor and use it for all similar projects in the future, which is a repeatable form of planning for multiple projects.

Example: Processes and workflows

Some deliverables, or goals linked to them, need certain steps that can become standardized and repeatable. For example, if the deliverables need to be designed individually and approved, you can prepare a simple workflow that makes all the steps, people involved, and approximate durations clear, so avoiding many difficulties. You should be careful, however, not to make the workflows and processes too complicated or too intensive in documentation, as it will have a negative consequence. All people involved in the project should see the workflows and processes as something that supports their work and makes everything easier for them, rather than as bureaucratic documentation that blocks their real work.

Agile projects have repeatable elements in their iterative development approach, where certain type of development activities are repeated for every feature; e.g. the common daily routine in XP (eXtreme Programming): pair up, pick an item, design it on a whiteboard, build the test scripts and code, integrate the code, etc.

Beside the repeatable workflows that can be used for technical activities, you can have repeatable elements for the project management activities as well. The processes in the PMBOK® Guide, PRINCE2®, and DSDM®, the activities in P3.express, and events in Scrum are examples of this concept.

Example: Cycles

Having repeatable elements for managing the project is helpful. This can be made even easier by putting them into repeatable cycles. These cycles significantly simplify the day-to-day activities of people involved in the

management and leadership of the project. Cycles of process groups in the PMBOK® Guide when used in a project with multiple phases, stages in PRINCE2®, daily, weekly, and monthly cycles in P3.express, iterations and timeboxes in DSDM®, and Sprints in Scrum are all examples of this concept.

Shorter cycles are easier to understand and use than longer ones; e.g., Sprints in Scrum in contrast to the phases according to the PMBOK® Guide. However, cycles that are too short may not be enough to support certain types of project, and the solution can be the use of multiple cycles, such as DSDM®'s use of short timebox cycles along with longer iteration cycles, or P3.express' use of daily, weekly, and monthly cycles.

Example: Methods

Using a methodology or a framework for running a project is another use of repeatable elements. This can be an existing system such as PRINCE2®, P3.express, DSDM®, or Scrum, or one that you've customized or built yourself. However, it's normally a better idea to start with one of the existing methods and adapt it to your needs than to build it from scratch.

Any repeatable element is abstract and needs customization to adapt it to the real world. There's a spectrum of abstraction and need for customization, though: small, relatively concrete quality checklists are at one end of the spectrum with the least amount of abstraction and need for tailoring, while methodologies are at the other end, with the highest need for tailoring. You should always note the need for tailoring, otherwise, the repeatable element won't match your needs properly.