**Constrains in Software Development**

### Introduction

For many years project managers have been encouraged to look to the Triple Constraints to provide a framework to plan, monitor and control a project. In its Glossary,***A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*** defines the Triple Constraint as “a framework for evaluating competing demands.” (PMI, 2004, p. 378) These Triple Constraints *(time, cost* and *scope*, with *quality* occasionally included as an adjunct to or substitute for scope, or as a fourth constraint) indicated the key factors that both defined the framework of a project, and directed project managers as to where adjustments would have to be made if one or another of those constraints became problematic.

You could only request two of the three, and the third factor would get defined by the first two. The shorthand version of this model said “cheap, fast or good – choose two”. If you want to have it cheap (low cost), and fast (short time frame), you would have to settle for limited scope or lower quality. If you wanted it fast and good (high quality with full scope), then it would cost you. We also understood that if the project was expected to run longer than planned, to bring it back into the desired *time* frame would require a compromise on either *cost* (i.e., spend more money to get it done on time) or *scope/quality* (reduce the scope, or settle for lower quality in the form of reduced characteristics, or less quality checking).

In recent years there has been greater understanding of the factors impacting on a project; PRINCE2™ has identified these revised factors through its focus on Tolerances. While building on the core factors of time, cost and scope**, PRINCE2™** has added *quality* (as a distinct factor), along with *benefits* and *risk*– to produce *six* constraints.

PRINCE2™ employs “tolerances” – its term for these six constraints – as *key* project controls. They are dimensions of the project for which ranges of acceptability are defined, which are monitored to identify or anticipate when a plan has entered “problematic” or “exception” territory. They are needed and used at all three planning levels of a project – the *project* as a whole, any one *stage* or *phase* of the project, and at the detail *work package* level.

This paper will first define what each of these six constraint factors means, in both theoretical and practical terms. It will then discuss how to use them during a project, and finally, locate them in the *PMBOK® Guide.*

### Constraints

#### Time and Cost

These are considered the standard constraints. They are reflected in our estimates and presented as *ranges* (plus-or-minus). In PRINCE2™ terms, as long as we are operating (delivering our projects) inside that agreed range limit, we are considered on-target. Good project management practice requires us to provide ranges for these constraints – ranges representing the estimating uncertainties associated with a project's particular circumstances. (PRINCE2™ separates these *estimating* uncertainties from funds set aside for specific purposes: a Change Budget – a fund to implement *requests for change* to scope or quality characteristics, or a Contingency Budget – funds to respond to previously defined *risk contingencies*.)

Classically time and cost are the first place the sponsors look (and for some sponsors, the *only* place) to see if a project is “in trouble” – ie, not meeting stakeholder expectations. This is probably because they are the most *tangible* measurements: we have a due date and a budget – what could be simpler? For many on the sponsor level the “scope” and “quality” of the project are harder to grasp – and measure. The assumption very often is: “Of course it will do what it's supposed to. Isn't that what we're paying for? Make sure it works and get it done on time. I don't care what it takes, as long as it comes in within budget and on time!” It becomes easier to disguise some missing features, or gloss over quality checking. Since the sponsors usually aren't quite sure what the detail scope characteristics are, or what goes into “quality checking,” everyone is just as happy to see them ignored or minimized, in favor of time and cost.

In the classic model, Risk and Benefits constraints don't even exist, so they are certainly not under consideration. We will soon see the importance of the four other factors that make up the six constraints.

#### Scope

Scope doesn't have the same ease of definition – ie, as normally being defined through “ranges”. Scope refers to the particular deliverables (“products” in PRINCE2™ terminology), which have been agreed to by the project's owners. In most cases there are no “ranges of acceptability” for scope: we asked for particular items, and we expect to get them – neither more nor less. We *can*, however, represent scope as a range, and it might look like the following. I typically would ask the project manager to produce Item A. With “scope tolerance” I might say “I want Item A – that's a must. if you have enough time or money left (ie, you can afford it) I'd also *like* you to build me Item B. But if you can't, I can live with only getting A.” With this kind of scope tolerance, the project manager has *flexibility* on what is to be delivered, but within strict and agreed limits. Scope tolerance could also show up as a “negative” range as well: “I would like you to develop items C and D, but if you are running short on time or money, it's acceptable if you don't produce D”. Either of these situations might occur when there are essentials that the project *must* deliver, but there are other items which are discretionary, or can be delivered at a later date without compromising the project's key objectives.

#### Quality

The quality constraint (or “quality tolerance”) is actually quite similar to that of scope – except that quality focuses on *characteristics* of a deliverable. When we address quality we are not looking to add (or delete) a *new* item. We are only looking to *alter* or provide flexibility (or “breathing space”) for some feature of an *already-defined* item – or to assure that a particular characteristic is *present and working properly* (quality checking). Here's an example. If I am building you a custom-made table, and you say “I want the table only to be made from cherry wood,” then you have given me NO quality tolerance on that particular characteristic of the table. If, for some reason, I can't find cherry wood, then I *do not*have the flexibility to substitute anything else, *unless I ask your permission.* With “quality tolerance” you might say “I want the table made out of cherry, but if you can't find cherry, or it is very expensive (above a certain cost), then I give you permission, in advance, to substitute walnut or oak.” You have just provided me with quality tolerance. You are not adding a new item, just giving me flexibility on one of the item's characteristics. If I can't find good cherry wood, I can use oak *without getting your permission.*

Quality tolerance also operates in the realm of *exactitude*, which is a traditional measure for quality criteria. It represents the degree to which a developed item *matches* its defined characteristics. For example, if I ask for a table to be exactly 1 meter long, then if it is delivered at 99 cm or 101 cm it would be rejected. As an alternative I might have quality tolerance of +/- 1 cm. That means I could deliver a product measuring 99.5 cm or 100.8 cm, and it would still be acceptable, passing its quality criteria.

Quality works in the same mode as the other constraints. For example, if a project is running late or over budget, the project manager may still be able to deliver the expected *items* – but they might not be tested as thoroughly (ie, we do not assure that the characteristics are present and working properly – very common!), or some characteristics of that item may be *reduced* or *eliminated.* This is how quality operates as a constraint. Some models of the triple constraint triangle use quality instead of scope as the 3rd leg of the triangle. In many classic situations, when time or cost was strained, it was quality – usually through less testing or verification, but sometimes through dropped characteristics – that was compromised.

#### Benefits and Risk

The last two elements of the six-constraint model are the newest and least-familiar ones, and could be considered controversial – except that they are both already present in projects. We are not creating them – we are just bringing them to the forefront and demonstrating how they interact with the “classic” constraints (time, cost, scope, and quality). When these two new constraints – benefits and risk – are not considered, they are likely to be neglected and produce a negative impact on the project and the organization. We will examine those potential consequences as we discuss each of them.

#### Benefits

First let us look at benefits. Benefits represent the *value* the project is expected to deliver to the organization. In PRINCE2™ terms, we don't do a project just because someone “feels” like it, or (in vague terms) “thinks it's a good idea.” PRINCE2™ requires the project to have a Business Case – a *clear*justification, with *measurable*, agreed benefits that are expected to result from the project's outputs. If there is no clear justification, then *the project should not be started*, and if the justification disappears – or is reduced below an agreed-upon limit – the project should be *stopped.* (That “limit” will become our constraint.) As the project has *deliverables* that it produces, the benefits represent the *value* that those items are expected to have for the organization (in financial or other terms).

While a project's objectives may be to deliver a new sales system, its *value* would be in its ability to increase sales, or improve customer service. Either one of these we would want to *measure*, to determine whether the new sales system was worth the time and cost to create it. Even governmental or nonprofit (charitable) projects need to have a Business Case – some measurable means to focus the project, and to use to assess the effectiveness of the project. (It is rarely possible to assess all benefits *during* a project. In a PRINCE2™ environment we would regularly anticipate and assess *expected*Benefits, and have a clear plan for an assessment of *achieved* benefits sometime after the delivery of the project's outputs.)

The benefits are affected by factors both internal and external to the project. PRINCE2™ recognizes that even if we are on time, on budget (cost), and meeting scope and quality expectations, a change in circumstances may indicate that it is no longer *worthwhile* to continue the project (ie, the benefits have diminished or disappeared). Considering benefits means we focus on the key question: “Do we want to continue expending limited organizational resources on work that has no discernible value to the organization?”

Here are a few examples.

I am a company developing a new product for our market. We will need to grab 25% of the market for that particular product for our investment to pay off. At the beginning of the project it looks like we'll get even more. Halfway through our one-year effort we hear the news: our major competitor has announced (and will start selling within 2 months) a product that is clearly superior to ours. We can still produce our item (and bring it to market), but it's quite clear we won't come close to breaking even. According to PRINCE2™'s approach, we have fallen below our “benefits tolerance” – and this constraint is under threat. With the triple constraint model we might have looked for the time or cost going outside acceptable limits – but here it's *only* the benefits that are in danger. We can still deliver to the expected time, cost, scope and quality – but we need the awareness (and measurement) of benefits to assess the viability of the project. *The standard triple constraints are clearly inadequate to help us diagnose this problem, or its magnitude.*

In this particular situation, we could pour in more funds (cost) to enable the project to come in earlier (reduce time), and therefore have a better chance of being competitive and achieving some (or all) of our benefits. It is the consciousness of this benefits threat – as one of our constraints – that has us looking to the other constraints to adapt to a change in circumstances.

In this same situation, we might also choose to cancel the project: we would rather cut our losses now, since we won't be able to get the return on our investment that initially justified the project. A benefits threat could also have been triggered by internal issues, such as senior management of the organization choosing to get out of the business that the project's deliverables would represent. (The degree to which a project supports organizational strategy is another way to define a Business Case or benefits measurement.)

#### Hardware Requirements

#### Legacy System

A legacy system, in the context of computing, refers to outdated computer systems, programming languages or application software that are used instead of available upgraded versions.   
  
Legacy systems also may be associated with terminology or processes that are no longer applicable to current contexts or content, thus creating confusion. In theory, it would be great to be able to have immediate access to use the most advanced technology. But in reality, most organizations have legacy systems - to some extent. A legacy system may be problematic, due to compatibility issues, obsoletion or lack of security support.   
  
A legacy system is also known as a legacy platform.

A legacy system is not necessarily defined by age. Legacy may refer to lack of vendor support or a system's incapacity to meet organizational requirements. For example, a large mainframe may use a 64-bit Java, while a Linux platform might utilize code from the 1960s. Legacy conditions refer to a system's difficulty (or inability) to be maintained, supported or improved. A legacy system is usually incompatible with newly purchased systems.   
  
Legacy systems are high maintenance and may involve intricate patching and modifications. Porting techniques are often used for software adjustments or adaptation. Older hardware may require added compatibility layers to facilitate device functionality in incompatible environments.  
  
An organization might continue to use legacy systems for a wide range of reasons, such as the following:

* "If it ain't broke, don't fix it!" The system might work adequately.
* The system is complex, and documentation is poor. Simply defining scope can be difficult.
* A redesign is costly, due to complexity or monolithic architecture.

#### Organizational Policies

#### Risk

Everyone recognizes the Risk on a project needs to be addressed and managed. We can see that in any project there may be a level of risk that we are simply not willing to live with, or “tolerate.” That is our risk tolerance. Its simplest and most common expression is in examining the probability of significant risks occurring, their potential impact on the project if they do occur, and our degree of willingness to live with those potential consequences. Risk refers to “opportunities” as well as “threats,” and can be applied in a similar manner.

“Risk Tolerance” of stakeholders is presented as an important consideration in the PMBOK® Guide's Management of Risk knowledge area, but how one establishes “risk tolerance” is not defined. Here you will see a clear, functional definition. These concepts can be brought back to the PMBOK® Guide and applied there as well.

The basic Risk model looks like this: (See Exhibit 1.)

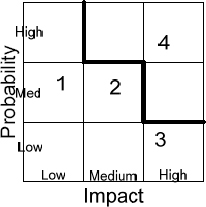


Exhibit 1

Each of the numbers represents an identified (and documented) significant project risk. We would be particularly concerned about threats that fall in the upper right-hand corner: medium-to-high likelihood of occurrence and medium-to-high (negative) impact of the project if they do. The more serious the threat the greater effort we make to mitigate that risk. The “risk tolerance line” – here indicated as the thick line – represents the *limit* as to the level of risk the project owners are willing to “live with” (or “tolerate”) for that project.

Here is an example. Our project is developing a new product for our domestic market. We have identified several significant risks, which have been mapped into our model (Exhibit 1). Risks #1 and 2 represent risks that are under control, and are not significant enough to stop the project. We will look at risks 3 and 4.

#### First Scenario

Risk #4 is a Supplier risk. We have a great supplier, but they are having financial problems. The probability that they will have trouble delivering looks *high* right now; the impact would be that we couldn't deliver our product. As of now we haven't come up with any satisfactory mitigating responses that would protect us. The risk tolerance of the project owners could be: “We can't live with Risk #4 as it stands. If there are no options to deal with it, it's better to stop the project right now before we commit too many resources.” This particular risk is on the wrong side of their “risk tolerance” limit (which *they* had established). They could also say: “Let us look for additional ways to mitigate it – like an alternative supplier. If we can have a reliable contingency plan, the probability would still be high, but we would reduce the impact (to the “probability high, impact low” box), so the overall rating would fall *below* our risk tolerance line.” (See the new location of risk #4 in Exhibit 2.) The project risks are now within their constraint/ tolerance limit.

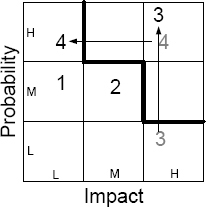


Exhibit 2

#### Second Scenario

Risk #3 is the risk that our market-dominant competitor might come out with a similar item. At this point that looks very unlikely (from what we know), but the impact on our project would be high if it *did* occur. (See risk #3 in Exhibit 1.)

Risk #3 is acceptable right now, since the probability is low, and it falls *below* the project owners’ risk tolerance line. If, during the project, we hear more reliable rumors that it is becoming *increasing*probable that our competitor will come out with their product, then risk #3 would move to “probability high” with the same level of impact. (See the movement of risk #3 in Exhibit 2.) That would put the risk *above* the risk tolerance limit – ie, “we cannot live with that level of risk – we have to re-consider our situation.”

There could be three choices for the project owners at this point, with risk #3:

a) find a mitigating response that will reduce probability and/or impact to an acceptable level;

b) close the project down, since the project owners are unwilling to live with projects that have any risks above their agreed-upon risk tolerance line;

c) *move the risk tolerance line* to some point above the new risk level. (See the new solid line representing the revised risk tolerance in figure 3.) They can move risk tolerance just as cost or time limits might be re-set or adjusted. “We decided reluctantly that we have to go ahead and live with a higher level of risk, because this project is critical to our survival.”

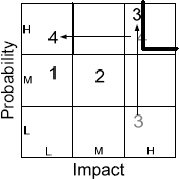


Exhibit 3

### How to Use the Six Constraints Model

We use the constraints model to help us *control* the project. Building on the examples we have provided above, the project manager with the sponsor/ stakeholders/ Project Board (Project Boards are how PRINCE2™ clarifies the sponsor role and responsibilities) establish what the project's (and phase/ stage) *limits* are in these six key areas. (Benefits and risk are usually defined down to the project and phase (stage) levels only; the other four are also scaled to the work package level, if required.) Together the six provide a *complete* set of guidelines for the project manager to know (a) what the sponsor/ stakeholders/ Project Board see to be important (since they have a vested interest in the project's success, and are providing the funding), and (b) what are their *limits* of acceptability in performance.

Constraints/ tolerances are set by the sponsor/ key stakeholders/ Project Board, so there are common, agreed expectations for a project. These constraints/ tolerances indicate for a project, as follows.

**Scope:** This is what the project is expected to deliver (if the scope tolerance is zero, the project must deliver no more and no less than what is specified).

**Quality:** The scope items are to be delivered with the defined characteristics (no deviation allowed if quality tolerance is zero), *and* all deliverables must be reliable – ie, the characteristics have been quality tested/ checked to the extent specified, so that they function as agreed..

**Time/ Cost:** The project manager is expected to deliver the final products/ deliverables within the agreed ranges. If the project is running late and/or costing more than the specified range, the sponsor/ stakeholders/ Project Board need to be informed, so they can decide how to proceed. If the project will end below the time or cost range, they *also* want to know, as other projects may be able to start earlier, or be able to use the unneeded funds.

**Benefits:** The sponsor/ stakeholders/ Project Board expect a minimal level of benefits to accrue from this project, or it may not be worthwhile to continue investing the agreed time and cost. (Even though we could deliver the agreed scope on time and within budget, to the expected level of quality, that does *not* mean the project is worth continuing. Delivered *scope* is **not** the same as *benefits*.) Similarly, if we see potential benefits running *higher* than our planned upper benefits tolerance range, the sponsor/ stakeholders/ Project Board may be willing to invest *more* time and money, or expand the scope, in order to assure (or take advantage of) those higher benefits.

**Risk:** We have agreement on the level of risk the sponsor/ stakeholders/ Project Board are willing to live with in the course of the project (their risk tolerance). If the project manager *cannot* control (mitigate/ transfer, etc.) major risks, then the sponsor/ stakeholders/ Project Board need to decide if they are willing to live with the greater risk exposure, or want the project to close down (“It's just too risky for us to move forward if we cannot mitigate, or don't have a contingency plan for risk ABC.”).

#### Balancing Constraints

Exceeding one or more of the constraint/tolerance limits is *not* a question of having done something “right” or “wrong”. It is an indicator that draws the attention of the project manager, to determine what has caused (or will cause) the deviation, and after analyzing possible consequences, devise a resolution coordinated with the sponsor/ stakeholders/ Project Board.

Well-thought-out constraints allow the sponsor/ stakeholders/ Project Board to “manage by exception” – ie, they allow the project manager to continue running the project, *as long as none of those constraints is anticipated to be exceeded.* If any of the six constraints has the potential to be exceeded, the project manager should approach the sponsor/ stakeholders/ Project Board to establish the cause and a course of corrective action (including possible cancellation of the project).

We have provided several examples of how an exceeding of one of the constraints can be balanced through examination/ extension of another constraint. In one case, a benefits threat (from a competitor's product coming out before ours) could be addressed by spending more money (cost) to get our product out earlier (time), or by taking greater risks, or by reducing or increasing the project's scope.

Here are a few other situations that the six constraint model would help to analyze and address.

a) Quality testing is on a tight schedule, generating greater risk that errors will not be detected and corrected. (This is a common situation – and presenting it in terms of “risk to the project” is more likely to get the attention of the sponsor/ stakeholders/ Project Board than just complaining.) There are additional implications, in that defective deliverables may threaten benefits as well. This could be addressed through additional time and/or funds to test properly.

b) Asking for additional scope or modification of characteristics (change requests) is addressed through additional funding and/or adding time to the project schedule. One would, or course, expect *benefits* to increase as well, and be worth the additional time/ cost.

c) A project is depending on the availability and functionality of an untested technology – a high risk situation. Allocating additional funds (cost) or time to confirm the reliability of the final deliverables may be a way of mitigating that risk. Alternatively, the company could reduce the risk by using a known technology, but that may cause the scope of the project to be reduced, or may reduce the expected benefits.

Notice the complex interactions between the constraints. Making changes – adding scope or quality – will (hopefully) increase benefits, but may also increase risk (that the changes may impact negatively on existing scope/ characteristics). Taking greater risks may increase benefits, but also have the potential to damage benefits if the risks don't work out. Taking more time may reduce risks (through better quality testing), or increase risk (if our product gets to market too late to compete effectively against other companies’ products, which would also reduce benefits). *The six constraints constantly influence and affect each other throughout the project.*

### Where the Six Constraint model fits into the *PMBOK® Guide*

Even though it defines the Triple Constraint as “a framework for “evaluating competing demands,” the *PMBOK® Guide* makes no specific reference as to how or when to *apply* the Triple Constraint to do just that.

It is clear, though, that application of the constraints/ tolerances occurs in 2 key points in the *PMBOK®Guide*'s process flow. The first is in **planning**, where we have to assess:

* what the constraints/ tolerances should be (if they have not been previously defined);
* who should be setting them (and when);
* how they are to be used by the project manager;
* how the sponsor/ stakeholders/ Project Board will be kept informed of the status of the constraints and project.

The second area is in **monitoring/ control**, where consideration needs to be given to:

* determining what is going on in the project (standard data collection/ monitoring processes);
* assessing how that compares against the constraints/ tolerances the sponsor/ stakeholders/ Project Board have agreed to with the project manager;
* whether any of the constraints/ tolerances been breeched – or threaten to be breeched; (PRINCE2™ emphasizes the importance of dealing with tolerance/ constraint breeches as soon as they are forecasted, rather than waiting for them to occur, so there are more options – and time – to deal with the situation.)
* proposing and recommending alternatives for addressing the breech.

The planning and setting of constraints/ tolerances should occur in each of the respective knowledge areas addressed by them: time (PMI, chapter 6, pp. 124-156), cost (chapter 7, pp. 157-178), risk (chapter 11, pp237-268), scope (chapter 5, pp103-122) and quality (chapter 8, pp. 179-198). Although benefits are not specifically identified in the *PMBOK® Guide*, they normally would be included in the Business Case section of the Project Charter (PMI, 2004, p. 45). The *PMBOK® Guide* does not make provision for regularly referencing the Business Case in the course of the project, so any ongoing use of benefits and the Business Case in assessing project viability will need to be added through the use of the required project management methodology (PMI, 2004, p. 95). (See below how the use of the PRINCE2™ project management methodology will assist in this.) How the constraints/ tolerances are to be used by the project manager and stakeholders/ Project Board needs to be part of the Project Management Plan, under each of the ‘management plan’ sections (schedule, cost, etc.), as well as change control, issue management and communications.

The day-to-day monitoring and control of the constraints occurs in “4.5 Monitor and Control Project Work,” (PMI, 2004, p94) where the key Tools/ Techniques are: “4.5.2.1 Project Management Methodology,” “4.5.2.2 Project Management Information System” and “4.5.2.4 Expert Judgment.” (p. 95) Your “expert judgment” now includes the understanding gained here about the value of using all six of these constraints to track the project's progress. It is also used to determine when the project is in trouble – ie, when one or more of the six constraints has exceeded, or is anticipated to exceed its limits, as agreed between the sponsor/ stakeholders/ Project Board and the project manager.

PRINCE2™ – a respected and widely used project management methodology – can largely fulfill the requirements of “4.5.2.1 Monitor and Control Project Work: Tools and Techniques – ***Project Management Methodology***”. PRINCE2™ defines and manages all six constraints/ tolerances during the project. It additionally addresses the Business Case by identifying its contents (including benefits) and the points during the project when the Business Case needs to examined, updated and evaluated, and proposes a Project Board to clearly define the sponsor's role and responsibilities.

References

PMI (2004) *A Guide to the Project Management Body of Knowledge (PMBOK® Guide).* Newtown Square, PA: Project Management Insititute.

Office of Government Commerce (1989) *Prince2™* Office of Government Commerce: Norwich, Norfolk, UK.

© 2007, Jay M. Siegelaub  
Originally published as a part of 2007 PMI Global Congress Proceedings – Atlanta, Georgia