

# Have it your way

Mark Prince, Andy Schneider

BT, No.1 Sovereign Street, Leeds, LS1 4BT

BJSS, 1<sup>st</sup> Floor Coronet House, Queens Street, Leeds LS1 4PW UK

Mark.prince@syntegra.com

[andrew.schneider@bjss.co.uk](mailto:andrew.schneider@bjss.co.uk)

**Abstract.** Everyone involved in the software development process makes decisions. Having appropriate decision support is one part of the process, but sometimes the greater challenge comes after making the decision: actually having the decision adopted and carried through. For Project Managers and Technical Leads, being able to get your decisions acted on gracefully and managing their outcome (good or bad) is one of the set of skills that belong in any manager or leads toolbox. This paper presents four patterns that capture the common practices necessary to have decisions adopted.

## Introduction

This paper represents part of a larger effort to capture a set of patterns to fill the knowledge gap that confronts the inexperienced and aspiring Team Leader and Project Manager. As such the target audience for the pattern and proto-pattern work is IT Software professionals that have had between 2 and 5 years industrial experience. Line Managers and Programme Managers are not the target audience for these patterns.

The paper presents a number of patterns that are generally applicable in organisations. However, the authors have chosen to target the explanations at situations Team Leads, Project Managers and Technical Leads find themselves in from time to time. This paper will use the word Lead to refer to the person in one of these three roles.

This specific set of patterns provides techniques for operating tactically in an environment where open, honest conversations in an atmosphere of mutual trust are difficult. They do not address how to turn around your dysfunctional relationships, but rather seek to provide some useful “sticking plaster”.

# The Patterns

## HARNESS

**Context** *Your team wish to make a technical enhancement that will be hard to sell as a standalone activity. Your manager is rightly focused on the product and not technology. You may have broached the subject with your management by now and are preparing in Step 2 of THREE STEPS.*

**Problem** **Management, unless technical enough to understand the benefits to the product, have little appetite to action items when they are given a purely technical justification. This is a problem, as you need to get some technical items actioned.**

Technological changes can be derived from many places. For example, CHANGE QUEUE[3] puts off the less critical housekeeping chores to get something else done, namely getting a product out of the door. Eventually this housekeeping needs to be done or the eventual delivery will be compromised.

You want to do this housekeeping, but the problem in most projects is there is always more work to do. It can be hard to convince management to pay for housekeeping or technological improvements to the code base when you could implement functionality that adds immediate value. If you ask the management for time to do the work and justify this with generic terms such as ‘it makes the system more maintainable’, it is likely they may reject your request. Many leads hide this type of activity in the plan, but a team without guidance might housekeeping that won’t be updated this iteration, effectively spending money where it isn’t needed.

*Therefore*

**Present an argument that harnesses the changes you wish to make to the cost of the functionality needed today.**

Instead of presenting a sequence of non-functional changes in a project plan for the management to (inevitably) remove, harness each change to either new functionality it will enable or speed of delivery of a specific activity. For example, instead of indicating that 5 man days effort is needed to improve rendering code, describe how the ABD display function required in this iteration will take 15 days unless the rendering code is improved. In this case, the ABD display functionality will take 7 days. This approach provides for “Harness” but ties its efficient delivery to specific enhancements. Tying changes to functionality also shows due diligence, i.e. if ABD display function is not implemented then the tidy up of the rendering code may not be strictly necessary. For some required changes the cost may need to be spread across a number of functional

units, in this case link the required change to all the relevant functional elements.

An alternative to linking changes to new functionality is to link changes to defect rates. Describe how many defects are normally fixed in a specific area and then describe quantitatively<sup>1</sup> how these defects may reduce or become quicker to fix if the improvement to the rendering code is undertaken. This type of approach is often referred to as “Management by Fact”, an approach many management consultants take.

Once applied you have a strong correlation between technical activities that you know need to be performed and their impact on the end result the management value. This makes it much easier to communicate the rationale for technical changes to the code base. This approach also provides a framework within which the development team can reason about how their changes help evolve the system in the directions required by the customer.

**Incorrect Application** It can be tempting to be creative about the links between required changes and new functionality/defect rates. If this creativity is discovered it will undermine the approach.

**Examples** *Major Corporation.* At the end of a project there were a number of key re-designs in the change queue but no buy-in from the management to enact these changes in the next release. For instance, the protocol used between the client and server used business keys rather than primary keys. This created problems when duplicate business keys existed. The team had got round this problem for ‘go-live’ by ensuring all business keys were unique. However the approach (hack) was fragile and vulnerable to the re-introduction of duplicate business keys. This was a very technical change and to obtain traction, the rework and new functionality were linked together.

**Related Patterns & Practices** *Tailor Made:* Linda Rising and Mary Lynn Manns describe how selling an ideas requires you tailor it to your specific audience.

*Refactoring:* Refactoring is a continual process of making small changes to the code on an ongoing basis. Refactoring doesn’t need the types of justification discussed above because it is an agreed part of the development process. However, refactoring is often slowed down or stopped during the end game of a project and technological changes build up, often in a CHANGE QUEUE[3].

---

<sup>1</sup> Of course, the numbers will be a guess, but at least some degree of approximation has been attempted. A rough guess is often better than no guess at all and many managers like numbers.

## THREE STEPS

**Context**      *You have an issue and potential solution. You have time to explore this with stakeholders/management and will need to do so to obtain buy-in.*

**Problem**      **You have to present arguments to the management and wish to obtain your managers focus so you can engage in informed debate and persuade your manager that your proposal is appropriate.**

You can present the arguments as described HARNESS *or* THE GOOD, THE BAD AND THE UGLY. You need to make sure your manager is ready focus their attention on you. You can be informal but approaching your manager in the corridor, when their mind is on other business, will make it hard to get your point across. You could book a meeting but the manager won't be prepared and the meeting will be spent getting your manager up to speed. Bear in mind that managers like things to go according to plan; they definitely do not like surprises. If you drop an idea on a manager 'out of the blue' it will probably not be received very well.

*Therefore*

**Lead your manager along a structured path to a decision; first introduce the idea, secondly organise a formal meeting and thirdly request a decision.**

**Step 1:** Find out if your boss has strong feelings about the subject area. When you are next in discussion with him or her raise the subject, listen for any strong feelings and, if none exist, suggest the topic of discussion maybe an issue that needs addressing. Avoid getting into too much detail; this keeps it conversational and non-threatening. This stage mentally prepares your manager, shows them there is an issue and that you are thinking about it. With any luck they may ask you to do something about it, but this is not always the case.

**Step 2:** Organise a formal meeting with your manager. Formality will ensure that your manager's attention is more likely to be focused on the subject under discussion. Prepare in advance and distributed a brief pre-read if the topic is complex. Once accepted you can move to the third step.

**Step 3:** In the meeting describe succinctly what the issues and arguments are. Listen to your manager and adapt your proposals on the fly if needed. Finally, ask for a decision. If the manager is not prepared to make a decision then re-iterate the points, you want to exit with a decision. If your manager is insistent on needed time to think then get agreement to have another meeting once he or she has thought about the issues. Arrange the date and time right then. This avoids you being fobbed off with "I'll think

about it”.

At the end of this process you will have set the agenda, engaged your manager or stakeholder and reached a decision. You will have avoided ‘frightening the horses’ by raising an issue that the manager had not heard of. The manager will have had time to understand the issue and its complexity, focused their attention on the solutions and provided either the decision or the support needed for the proposed actions.

**Incorrect Application** Following this pattern for trivial decisions will add too much latency to small decisions. Using this pattern in an emergency where a snap decision is required also introduces too many delays.

**Examples** *Change Queue.* At the end of a project there were a number of key re-designs in the change queue but no buy-in from the management to enact these changes in the next release. HARNES was being used. One of the authors informally met the programme manager in the corridor to discuss the issue of the change queue. He suggested a meeting. This was organised and the author presented the arguments suggested by HARNES to get some items on the change queue actioned.

**Related Patterns & Practices** Applying ideas from areas such as Neuro-Linguistic Programming[5], [10] can help make you a more effective communicator.

Influencing strategies such as THE GOOD, THE BAD AND THE UGLY and HARNES can be used in step 3 to improve the chances of you getting your own way.

## ENOUGH FAT

**Context**      *You've made a decision and have a plan to carry it out.*

**Problem**      **Your management always trim tasks from the plan, they do not feel they have added any value unless something is removed. You need to ensure your plans are resilient in the face of this pressure.**

It is a temptation of management to trim estimates, and their motivations for trimming can come from several angles: recognising that some tasks are unnecessary, being bullish, pleasing their management, fixed budget ceiling, being short sighted, or satisfying a bigger picture view at the expense of your work. Whichever motivation your management has for trimming your estimate, you still need to ensure your plan can be delivered despite their improvements.

*Therefore*

**Ensure they your plan has enough fat such that it can be trimmed and still deliver.**

Put items into your plan that you can trade with if your management are going to remove items. There are several techniques (explained below) that will help with this. However, it's important for you to be able to give ground and have fat removed, as well as clearly identify which work must stay in (the meat).

In order to protect the plan you have, or put enough fat into your plan, consider the following:

- Apply contingency to tasks. You can do this explicitly to buffer tasks on the critical path or dependant on tasks on the critical path. It is best to add the contingency explicitly before major milestones rather than pad individual tasks. If you take this approach be sure to add in additional contingency so you can negotiated down.
- Plan to execute full test cycles (which is probably what your quality plan says anyway), but know you can maintain assured levels of testing with a targeted regression subset of tests. This means you can afford to give in the testing if necessary.
- Put Firebreaks into your plans. These are parts of the plan with no development activity, and are ideally placed near the end of the critical-path for particular task. They are highly visible and do give good warnings when a plan is over-running. If you have planned well, you should be able to tolerate the loss of your firebreaks.

- Weekends. Do not include weekends on your plans or as part of your estimates as this removes some ‘fat’ in your plan (they may be tempted to assume they are business as usual).

In all of these activities, it is important to ensure you are making your management aware of any increased risk introduced by removing activities or compressing timescales. There can be an incumbent assumption in management that development will pass through without incurring problems, and worse, stripping out any slack will somehow not increase the chances of problems occurring. Cover your back.

If your management are intransigent, and are insisting on compromising your plans even though your arguments are good and your plans are well formed, try to understand *their* motivations. It may be that there is a bigger game afoot, and you have to sacrifice your quality for this. Find out why the fat has been removed despite everyone’s good arguments, and explain it to your team. It will at least give them some consolation, and enable them to understand why they are working the way they are.

**Incorrect Application** The authors have seen three incorrect or over applications of ENOUGH FAT:

- *Blatant padding*. Just adding a blanket amount of time over the entire piece is the crudest way of putting time into your plan, and doesn’t leave obvious targets for your management to trim. Worse case is they don’t trust your planning or arguments.
- *Contingency applied on contingency*. Being over skilful in adding contingency means other people in the chain may not find it. This means they also may add contingency on your behalf. The worst situation the authors came across was a genuine 3-day task growing out to 11 days in the final plans.
- *Management don’t see contingency*. Any sensible plan has open as well as hidden contingency. If management do not think you are planning contingency in, they may question your planning abilities, which will bring your carefully added fat into question. Worse, they may add their own, which gives rise to the *contingency applied on contingency* problem outlined above.

**Examples** *Infrastructure Rollout*. One of the authors was under considerable time pressure to roll put a global infrastructure, and the programme was already strained for budget and working to political milestones. The author had plans accepted with formal lead-times for kit delivery, but had arranged a *just-in-time* agreement with the supplier. This meant that that the formal lead time was actually there as contingency – all planned contingency had been removed because the supplier was contracted by delivery SLAs, and as such no slip was expected.

**Related Patterns & Practices**

## THE GOOD, THE BAD AND THE UGLY

**Context** *You wish to obtain approval from management for a particular course of action. You know your manager and recognise they wish to add value and assist in the decision making process. Your manager is non-technical or their technical days are long gone but you need to ensure the right decision is made. You may be in Step 2 of THREE STEPS.*

**Problem** **You believe your management may exhibit too much of a bias, usually towards inexpensive tactical solutions, to make an effective decision in the area you wish to address.**

In many environments it can be hard for a technical person to convince non-technical management of the veracity of their approach. Management often wish to intervene and add their perspective, but in doing so often miss the subtleties the technical person is trying to communicate. The result is that decision can be rejected without an appropriate degree of discussion and consideration.

*Therefore*

**Construct a range of options that express a number of views but place your preferred option as the “obvious candidate”.**

Pick up a 750g weight, feel the weight, set it down and repeat for a 500g weight. You will note the latter is lighter. However, if you repeat but use a 10kg rather than 750g weight you will feel the 500g weight is lighter than the first 500g weight. The relative weights influence your perceptions. This alteration of perception by contrast is known to effect decision making [1], [2]. Most people, when presented with a decision, would prefer to see options and to know that alternatives have been considered. These two facts can be used to help you obtain the decision you want from management and ensure you consider alternatives. Consider presenting a range of options. Do not construct this range so all are equal in merit.

Instead, present four options<sup>2</sup> as follows:

1. The **ideal** solution. This may not be the most practical in the circumstances but it will definitely be the best in the long run. It brokers no compromise and you know it is highly unlikely to be adopted.
2. A **good** solution. This accepts trade offs need to be made, balances the long and short term goals and is the option you would like to

---

<sup>2</sup> Four options is a manageable number. Two options only allow a black and white interpretation of the solution space. If you add a third option then the ‘middle ground’ is easily identified as the decision you want. More than four options make it difficult to find alternatives with enough ‘clear water’ between them. If you exceed around 7 options then you may be exceeding the number of discrete chunks that can normally be held in short term memory[6], thus making the conversation harder. In addition, the more options you present, the longer it will take to arrive a decision.

- get adopted.
3. An **ugly** solution. This is more tactical than the **good** solution but still contains the core elements you wish to get adopted.
  4. The **bad** solution. This solution is very tactical, shows no compromise or acknowledgement of wider influences and, whilst seen to be in some sense valid, is obviously extreme and very unlikely to be adopted.

This approach allows you to present a range of options and show a degree of due diligence. A side effect of producing the list above is the approach forces you to construct different options and, as in design, considering multiple options is good practice. During construction of the options you may even find your view changes and there are other more credible alternatives.

Once the options are presented the management are very likely to dismiss the ideal solution as idealistic and the bad solution as unworkable. This leaves them a choice between two options. You favour both of these options. Ideally they will choose the good solution, but even if they choose the ugly solution you will have got your way. At this juncture you will have hopefully got your way. However, you find that the manager chooses the wrong option. If this is the case you have a number of options:

- Go ahead and 'do the right thing'. This works effectively if you can expect your manager not to check back later and fire for not following the agreed approach.
- Re-fashion your options and try again. This can be tough as it is often difficult to find a way to spin the options that isn't obvious to the manager. In addition the manager may not wish to spend more time on the subject.
- Wait for project to hit problems and then approach the management again. This is often the most successful approach but requires you watch carefully for warning signs and act quickly before the project gets into any significant trouble.

Either way, you have constructed and evaluated a number of options, shown due diligence and will have a clear conscience if a poor choice causes problems in the project.

**Incorrect Application** The ideal and bad options have to have some credibility, there has to be an explainable chain of reasoning to back them up. If they are too extreme then this will undermine your credibility.

If the gaps between the options are too small then the management may choose any of them or consider there are no real alternatives and reject the suggestions.

**Examples** *Change Queue*. A lead had implemented the CHANGE QUEUE[3] and needed to get some of the changes rolled into the next release. The lead presented four options:

1. Make none of the changes.
2. Roll some of the changes into ongoing development on an ad-hoc basis.
3. Select changes based on impact on new functionality.

4. Make all the changes before development of new functionality started.

In this case options 2, 3 or 4 were acceptable to the lead, but he knew that 4 was a non-starter.

*Web Services.* A large corporation has a standard that requires all integration to be over a specific message bus and that messages conform to a set of enterprise standards. Each message has to be agreed by a governance body. In this organisation the overhead of a message bus deployment and a governed message design was too expensive for a small project to fund. Many tactical projects therefore avoided the standards, seeking approval for non-compliance. The enterprise architecture group was trying to move the organisation to service based architecture and so provided two additional alternatives:

1. Wrap the legacy system in a web service.
2. Wrap the legacy system in a web service and model the request and reply content using the governance process.

This gave four options: the very short term 'seek non-compliance', acknowledge the direction of the architecture team in varying degrees (the two options above) or adopt the fully compliant architecture. Given one was unacceptable to the organisation and the last was too expensive for small projects, each tactical project would be steered in the direction the architecture team desired.

**Related  
Patterns &  
Practices**

Whitehead describes a similar model in [4] except he suggests ordering the options in "ascending impact of risk".

## RAISE THE YELLOW FLAG<sup>3</sup>

**Context**      *You've made your decisions, had them accepted, had your plans accepted to carry your decisions through and have embarked on your course of action.*

**Problem**      **Your plans are going wrong, what do you do?**

Despite your best intentions, schedules are running late, or the decision you have made is not paying off. You have convinced management to go your way, and irrespective of any correcting action, need to ensure they still support you.

*Therefore*

**Inform your management that things are going awry, and that you have plans to pull them back.**

Before you start on your course of action, find a way of baselining where you are, so that you can ensure that your new course of action is taking you in the right direction relative to where you were. Regularly collect metrics so you can be sure the course you are taking is still going to get you to your destination in time.

Ensure you have made known a set of stop of warning conditions (or even hard stops) based on the evidence you gather and use these as triggers to raise the yellow flag to warn management that things are running into problems and perhaps a new option is required (possibly go through another iteration of THE GOOD, THE BAD AND THE UGLY).

**Incorrect Application**      Take your warning to your management along with plans to remediate. Lack of sensible data on which you make your decision leads you to call off or modify an activity unnecessarily. It may be difficult to make proper decisions about a course of action and make a mid course correction if you haven't been gathering the correct data from the outset.

Crying Wolf [9] will mean that you start to set the impression that you are not managing the choices you have had accepted. Be sure you can call off a choice only when you can prove that course of action will be dangerous or will not deliver to the acceptance criteria you are working to. Repeated

---

<sup>3</sup> From the motor-racing convention when a yellow flag is raised, warning drivers that an unseen hazard is ahead and they must obey a set of rules until the flag is lowered.

application of SILVER BULLET can lead to Crying Wolf.

**Examples** *Increased Team Throughput.* A lead had two choices acceptable to management to deliver a piece of work: extend the deadlines, or ask the team to work repeated long hours. The latter was least desirable to the lead, but extending deadlines was regarded as politically unacceptable. Before putting the team on longer hours, the lead gathered productivity and Defect Density metrics and noted these against the effort to fix defects. This data was trended over time. As suspected, there were initial rises in productivity, but then the team started to become tired. The productivity started to drop, and the rate of defects introduced started to rise. Once these trends were established, the lead RAISED THE YELLOW FLAG to management, a new iteration of THE GOOD, THE BAD AND THE UGLY was executed, and this time, the decision to extend the deadline was accepted.

**Related  
Patterns &  
Practices** SILVER BULLET [8]

## Conclusion

Whilst not a comprehensive set of patterns around decisions and influencing, the paper has provided a small set of patterns that will hopefully allow you the reader to get your decisions adopted and protect the decisions that are important to you. Ideally you should not need many of these patterns, as your communication with your management should be open, honest and in an atmosphere of mutual trust. If this isn't the case, then whilst you are working on creating that atmosphere of trust, the authors hope patterns are useful.

## References

- [1] *Harvard Business Review on Decision Making*. Harvard Business School Press 2001.
- [2] Huczynski, Andrzej & Buchanan, David. *Organizational Behaviour, An Introductory Text*. Prentice Hall 2001.
- [3] Prince, Mark & Schneider, Andrew. *Patterns for the End Game*. EuroPLOP 2004.
- [4] Whitehead, Richard. *Leading a Software Development Team*. Addison-Wesley 2001.
- [5] Knight, Sue. *NLP at Work*, Nicholas Brealey Publishing 2002.
- [6] Miller, George. *The Magical Number Seven, Plus or Minus Two*. Psychological Review 1956, vol. 63, pp 81-97. Reproduced at <http://www.well.com/user/smalin/miller.html>.
- [7] Rising, Linda & Mans, Mary Lynn. *Fearless Change: Patterns for introducing new ideas*. Addison-Wesley 2004.
- [8] Fred Brookes. *No Silver Bullet: Essence and Accidents of Software Engineering*, IEEE Computer 1987. <http://www.computer.org/computer/homepage/misc/Brooks/>.
- [9] Aesop, "The Shepherd boy and the wolf", Aesop's Fables.
- [10] O'Connor, Joseph & Seymour, John. *Introducing NLP*. Element, 2002