Table of contents

Welcome 2
A mythical software project #1 2
Six thinking hats 4
What each hat represents 4
Blue hat 4
White hat 5
Yellow hat 5
Red hat 6
Black hat 6
Green hat 6
More about hats 6
Improving our working relationships 7
Using the six-thinking-hats in reviews 8
Using the six-thinking-hats to design test cases 8
With the hats we can ask questions such as 9
A mythical software project #2 10
Using the six-thinking-hats to assess the product 12
A case study on using the six thinking hats for UAT 12
Using hats to discuss release meetings 13
More hats for software testing 13
Mapping testing ideas to 'testing-hats' 14
Other ways to improve your testing 16
Risks of using hats 16
Feedback from practical experiences of using the six-thinking-hats 17
What else do we need to consider? 17
Conclusions 17
The final task 17
References and further reading 18
Acknowledgments 18

Welcome

Welcome to this interactive tutorial on six hats for software testers. You are an intrinsic part of the learning process. We have a number of practical group exercises, and with your active participation I hope you will "learn by doing".

A mythical software project #1

For the last six months the company has been working on the next version of their online widget site. Budgets are tight and the CEO insists the software needs to go live within the next three weeks, otherwise he may have to cancel the planned bonuses and he may even have to cut headcount. You are part of the project team. The software is still incomplete and nowhere near ready to go live.

You are about to join a project meeting to decide what needs to be done over the next week in order to increase the chances of delivering working software within the timescales specified by the CEO.

What happens next? Please make notes on the next page.

Your notes	

Six thinking hats

Edward De Bono's 'six thinking hats' has been used across the world over the last 20+ years by large numbers of people. The concept of 'six hats' is highly praised by proponents who have applied them to solve business issues.

The hats provide a metaphor for six different and distinct viewpoints. Each hat has a color associated with it, rather than a complex style or name. By having simple, plain colors the hats are easy to recognize and understand, in many countries of the world.

The six hats are:

Hat	Rationale	Comments
Blue	Control and organization	Provides focus. Helps to monitor progress.
White	Factual viewpoint	Concentrates on obtaining and finding facts and figures
Yellow	Positive and speculative	Constructive thinking, encouraging
Red	Emotions and feelings	Intuitions and hunches
Black	Cautious and careful	Concentrate on what can go wrong, protective
Green	Creative thinking	Providing fresh alternatives, without criticism

The hats may be physical, or imaginary. Regardless, they help us to quickly set or change the direction of our thoughts. Their main purpose is to help us direct our thoughts and ideas in a particular direction. Hats can be swapped quickly, sometimes once a minute, when working alone, or in small, cohesive teams.

What each hat represents

Note: the following descriptions are a summary of those provided by Edward De Bono in his 'six thinking hats' book. His work is the standard reference. Details of the book are provided in the references section, later in this document.

Blue hat

Blue represents the blue sky above, and helps to provide an overview. With the blue hat we:

- Think about our thinking
- Define the purpose e.g. of a meeting or a task
- Controls the use of other hats
- Gather the outcome, at the end of the meeting
- Set out the next steps

In a group the blue hat may be worn permanently by the facilitator. When the group uses the blue hat individuals can make procedural suggestions e.g. on which hat to use next.

Participants may also use the blue hat to clarify points of procedure e.g." Is that a white hat comment or a red hat comment?"

White hat

White represents a sheet of paper, perhaps a computer printout. The white hat is about information. The information can range from hard facts and figures to things we believe but don't yet *know*. The information may include stating second-hand facts e.g. "The CEO of our competition claims their product can sort widgets by color and texture", or stating other people's beliefs or feelings e.g. "Jack is angry with the project team and blames the testers for holding up the launch".

Sometimes the information may be contradictory, if so we record everything, and only decide later, if and when we need to make a choice.

The white hat is neutral, and reports on the world as we find it. With the white hats we ask questions such as:

- What information do we have?
- What information do we need?
- What information is missing and what questions do we need to ask to get the information we need?

We should question 'facts' because beliefs and opinions may masquerade as facts unless they are challenged and double-checked. Using the white hat we try to qualify facts to determine our degree of confidence in the information presented. For instance we may use a scale of: Certain, Fairly confident, 50/50, Uncertain, No chance! – pick a scale to suit your project and circumstances.

Yellow hat

Yellow represents sunshine, optimism. The yellow hat challenges us to find benefits, and to see whether it's possible to put an idea into practice. According to De Bono people sometimes struggle to recognize the benefits in their own ideas, so the group may need to help to nurture other people's ideas when using the yellow hat.

The yellow hat is deliberately positive. It's about actively seeking positive benefits. It may be very effective at shaping ideas spawned when using the green hat e.g. we use the yellow hat to answer the question: "Which of these creative ideas can we apply, and what would the benefits be?"

Red hat

Red represents anger, passion, love, fire! – what fires us up, energises us, annoys us, upsets us, etc? The red hat provides a safe, controlled vent for people to say how they feel. These feelings may be positive or negative. Without the red hat, people's feelings may be masked and appear as facts or opinions "it'll never work here because no-one wants it…" And once the red hat is removed we can put our emotions to one side and focus on other directions.

We can use the red hat to ask people how they feel about something e.g. about our plan.

The red had includes intuition, hunches, and emotions. You do **not** have to justify your contributions!

Black hat

Black represents darkness, and allows us to be careful and cautious (without being overly negative). The black hat helps us to identify things that might be incorrect, stuff that might go wrong, etc. It provides a natural caution, which is particularly useful in software testing ☺

The black hat taps into our natural 'fears' about safety and security and helps us to voice our concerns without them being viewed as 'negative' or 'unhelpful'.

Green hat

Green represents fertility, new growth. With the green hat we seek new ideas, alternatives, and fresh inputs. We may use lateral thinking (another term coined by Edward De Bono) to gain fresh insights to a current issue or problem. When wearing the green hat we may use brainstorming to come up with new ideas. These ideas should not be criticized when wearing the green hat.

Sometimes ideas that seem silly when we first encounter them become key drivers e.g. "We will pay our customers to spend money with us" has led to loyalty schemes for a vast range of industries.

More about hats

Although we might be tempted to categorize someone as 'emotional' or 'negative' we need to resist the urge to do so, particularly when using the hats. Rather, we need to encourage people to try out each hat so they can contribute more fully to the goals and objectives the team are trying to achieve. Everyone needs to be able to use and apply each, and all, of the hats.

We can use the hats individually, or in groups. When used in groups every member of the group uses the same color, at once, before the group switches to

another color. By using a common color the group can be much more productive, rather than having in-fighting.

We can use the hats throughout the software industry, including:

- Improving our working relationships with other members of the team, including the developers, project managers, etc.
- Reviewing documents, code, etc. (static testing)
- When we design test cases
- When we execute the tests
- Providing usability feedback

The hats help provide a simple, and easy-to-apply, framework to help us to improve the quality of what we do and of the software we provide.

Improving our working relationships

Software development practices may be viewed as adversarial for instance when software is 'thrown over the wall', or when groups blame each other for problems and delays. For instance: 'If you'd given us better specs, we wouldn't have had to spend an extra 3 weeks and $50,000 trying to find out what the users needed…'

By adopting the principles of the six-thinking-hats we are able to disentangle facts from emotions (white and red hats), to present careful and cautious views (black hat) balanced by a mix of positive thinking (yellow hat) and creative suggestions to improve things (green hat). And with our blue hats we can consider the 'bigger picture' of how our work fits with the project and company objectives. If the other people we are working with don't know about the hats we can choose to use more neutral terms to describe our ideas and thoughts in each area.

As others come to see the benefits of our work we have the opportunity to explain how we managed to improve our outlook and our work by using the hats. Hopefully others will start to use the hats as well, which will help to further improve the communications within the team.

Using the six-thinking-hats in reviews

A key skill is the ability to review artefacts related to software. Everybody who's involved in software has to review material of one sort or another, ranging from rough requirements documents to thousands of lines of source code, to the test cases, etc. Sadly, we are often less effective than we'd like, and our feedback may be perceived as unhelpful at best. However by using the six-thinking-hats we have an opportunity to improve the effectiveness and quality of our reviews.

Each hat allows us to view the material being reviewed from six distinct directions. And as we can categorize our comments and feedback under the various hats we can reduce the perceived personal 'attack' on whoever's responsible for the material.

Using the six-thinking-hats to design test cases

Generally our testing fits within a larger context, that of shipping working, desirable software cost-effectively and on time. One well accepted practice is to incorporate risk in order to decide what to test, how to test, and how much to test. The practice includes risk analysis and test design, and is known as "Risk-based testing".

Risk-based testing is a key driver in deciding what to test, how to design the test cases, and to decide how much testing will be sufficient to balance the risks with the rewards. However, our testing is only as good as our understanding of the risks. In some projects the risks aren't consciously considered or addressed. In other projects risks are formally captured and analysed very early in the software lifecycle.

One way to improve our understanding of the risks, and then to design appropriate tests is to use the six thinking hats throughout the process. Each hat is used to help improve the effectiveness and appropriateness of the resulting tests. The following table provides examples of questions we created by viewing the problem from the perspective of each hat.

With the hats we can ask questions such as

Questions	Hat
• What data do I need to design this test case?	White
• What results do I need to collect?	White
• What are the business objectives and requirements?	White
• What advantages do we obtain from designing (or skipping) this test case?	Yellow
• What outcome would we like to achieve?	Yellow
• How do I *feel* about the software being tested?	Red
• What sort of things would annoy me if it didn't work properly?	Red
• What would annoy the users or the customer if it didn't work properly?	Red
• What sort of problems could go wrong with the test? What sort of things might we get wrong, or misinterpret?	Black
• How can we design our test cases to reduce the chances of the tests being inappropriate or problematic?	Black
• What business or economic (money) impact would there be if the software didn't work properly?	Black
• Are there novel ways we can test the software (e.g. using exploratory testing techniques to complement scripted testing)?	Green
• Are there alternatives to the way we currently do our testing that might improve the results, timescales, costs, etc. of our testing?	Green
• What are we trying to achieve with our tests?	Blue
• When will we know when we're done?	Blue

Let's see whether they can help us increase the chances of shipping the software.

A mythical software project #2

For the last six months the company has been working on the next version of their online widget site. Budgets are tight and the CEO insists the software needs to go live within the next three weeks, otherwise he may have to cancel the planned bonuses and he may even have to cut headcount. You are part of the project team. The software is still incomplete and nowhere near ready to go live.

You are about to join a project meeting to decide what needs to be done over the next week in order to increase the chances of delivering working software within the timescales specified by the CEO.

This time: use the *six thinking hats* during the meeting. Focus on the sorts of testing to do, however feel free to make notes about other aspects of the software, such as usability, etc.

What happens next? Please make notes in the relevant boxes on the next page.

De Bono's thinking hats	
Green Hat Creative approach Fresh new ideas 	**Blue Hat** What have we learnt? What are the next steps? 
Red Hat Gut feelings What do my senses tell me about this? 	**Yellow Hat** Advantages? The best possible outcome? 
Black Hat Risks and problems? Worst case scenario? 	**White Hat** What facts do we need? How can we get those facts? 

Using the six-thinking-hats to assess the product

The method can be used to assess the software being developed, e.g. to identify potential problems, missing functionality, etc. The following brief case study provides an example where the project team reviewed some new software prior to accepting it from their supplier.

A case study on using the six thinking hats for UAT

A small software testing consultancy commissioned a new version of their web site, which was being developed by another company. The project team were responsible for user acceptance testing (UAT). They used De Bono's six thinking hats as part of the UAT process. The initial review, including an introduction to the 'hats' took an hour.

Each member of the team used a template similar to the one we used for the exercise on the previous page. As they reviewed the current version of the web site, they were encouraged to make notes under the various hats. At the end of the meeting the notes were collated and analyzed. The notes covered various key aspects of the site, including:

- Core functionality: e.g. how to update the content
- Usability: including page size, aesthetics, look and feel, etc.
- Privacy and Security: e.g. who can access information, how well is the client data protected, etc.
- Performance: although only in general terms
- Process testing: including the interaction between people and the functionality offered by the web site – who does what, and when…

As a result of the feedback, the launch of the new site was delayed until various changes were made.

Using hats to discuss release meetings

Another consultancy provided examples of how they used the six-thinking-hats to discuss release decisions – should this software go live, or not?

The hats were applied on key findings that were the subject of discussion, for example – defects not repaired yet – and helped to provide fresh insight into whether the software was likely to meet the release criteria.

More hats for software testing

Debates have raged within the software testing community about what is and is not software testing and how to perform software testing effectively. Some people claim we cannot and should not test without requirements, others claim that scripted testing is futile and should be replaced by exploratory testing, for example. One important contributor to the various approaches to software testing is Bret Pettichord, who describes 'schools of testing'. In his original work he identified 4 schools:

- The analytical school: where we take an analytical approach to testing
- The quality school: where testers are the 'quality police' who stop the developers from shipping 'bad code'
- The factory school: which is process-driven to reduce the costs and risks
- The context-driven school: where the testing is based on the current context and circumstances

Subsequent work suggests a fifth school, using test-driven development practices. Test-driven development encourages unit tests to be created before the code is written. The unit tests are then run to demonstrate the software works correctly. A number of variations exist on the concept of putting the tests first.

Sticking to a single school is risky and may artificially limit the effectiveness and productiveness of the testing. Various testing practitioners agree a mix of schools will help to improve the quality of our work.

Exploratory testing combines test case design with test execution, where the next test is often based on the experience of the immediately preceding testing. According to proponents it offers tremendous opportunities to find new bugs, by not being limited to, or constrained by, scripted tests.

And risk-based testing, as mentioned earlier, is becoming more and more popular as a way to decide where, how and how much to test.

Finally in this section, benefits-driven testing tests the software to demonstrate whether specific benefits have been achieved. For example one of the key benefits for the widget site is to implement an online payment module which is expected to deliver significant benefits to the business, by allowing us to easily process orders from new customers who might not want to create a business account with us. Therefore we might want to test the new payments module as a higher priority as it is expected to deliver a key and immediate benefit to the business.

Mapping testing ideas to 'testing-hats'

We can map these various ideas to create 'testing hats' that combine the concepts proposed by Edward De Bono with some of these ideas from the software testing community. The goal is to help us to improve the ways we test software, by giving us at least six different ways to approach our testing. The overall outcome provides a powerful way to improve our testing!

Testing idea	**Hat**	**Comments**
Benefit-driven testing	Yellow	Looking to show the benefits of the software being tested, e.g. showing what works, rather than what is broken.
Risk-based testing	Yellow (and black)	Only test to assess or mitigate risks related to the project.
Factory school	Blue	Testing is only part of a bigger picture and needs: • To match the business drivers, • Integrate with other tasks, • To fulfil the project and other objectives.
Quality school	Black	Process-oriented, leave little to chance: for instance we may want to implement regression testing to reduce the chances of old bugs reappearing.
Analytical school	White	Also includes white-box testing techniques, and focuses on applying proven techniques.
Context-driven school	Green	The tests are based on the context, which tends to be an iterative, evolving, creative process.
Test-first / test-driven development	Red	Commit to testing early, before the main software is written, passionate about testing. Some practitioners are called 'test-infected'.
Exploratory testing	Green (and Red)	Creative, seeking alternatives and fresh ideas for finding bugs.

From the previous table, which maps the various testing ideas to the colors of the hats used by De Bono, we can now reassess our software testing, and hopefully find additional ways to look for bugs that may have hidden from us in the past.

You can even use the six-thinking-hats to explore the application of each of the approaches mentioned in this section, for instance – what are the advantages, and risks with using test-driven development? By applying each of the hats in turn you may decide the effort in implementing one or more of the schools is worthwhile.

Don't be constrained by the mapping presented here, you may want to adapt the model to suit your needs and interpretation. What's more important than picking a particular color (or colors) is for you to gain fresh insights and perspectives on how to test effectively and efficiently.

Other ways to improve your testing

Testing of software functionality is already tough, and testing of the qualities and constraints introduced by things like performance, security, usability, etc. is even more challenging. These qualities and constraints are often described as non-functional qualities, or requirements, and the testing is called non-functional testing. One way to consider these non-functional aspects is to again map particular non-functional aspects to hat colors, for instance:

Non-functional attribute	**Hat**	**Comments**
Accessibility	Yellow, green	How can we ensure as many people as practical can use the software, regardless of who they are, their physical and mental abilities, etc?
Performance	White	Performance is all about numbers, how fast? How many? Mow much (memory, CPU, etc)?
Reliability	White	How many failures do we find when the software is used under typical operating conditions?
Recoverability	Black, yellow	What happens when the software fails? Do the automated and manual recovery procedures work?
Security	Black	What are the limitations and weaknesses of the software?
Usability	Red	How do people *feel* about using the software?

Risks of using hats

Although the concepts of using the hats are simple, we can make mistakes or misunderstand how to apply them appropriately. Common mistakes include:

- People sometimes present opinions as 'facts' during white hat thinking
- Hats can get mixed-up e.g. confusing positive thinking , where we seek the best-case positive outcome (yellow), and creative thinking (green)
- The blue hat does not get sufficient attention. As a result meetings may lose direction, and lack a conclusion.
- People may not obey the rules, e.g. they may get into arguments and start to defend their ideas, rather than focusing on getting their ideas 'out'.
- People may not take part in the discussion. The facilitator should ensure that everyone is offered an opportunity to voice their ideas.

Feedback from practical experiences of using the six-thinking-hats

Using the blue hat to frame the meeting will help to address some of the risks. At the start of the session the blue hat is used to agree on specific objectives the group is aiming to achieve. Because the hats encourage parallel thinking, using hats does not always lead to conclusions or actions. You still need to decide what needs to happen next. The blue hat should also be used at the end of the session to form the conclusions and to identify the follow on activities.

What else do we need to consider?

Using hats may change our viewpoint of the software in other ways, for instance 'yellow hat' thinking may lead to:

- Faults becoming 'features'
- The test results not being checked properly (as the tester wants the tests to pass)
- Testers to continue testing, even when they haven't found any faults recently (as the next fault is 'just around the corner!)

The hats are only one set of tools, there are many more that may be useful to you. For instance: perspective-based reading (PBR) is gaining credence as a way to improve the effectiveness and cost-effectiveness of inspections. However it differs in its approach because unlike the six-thinking hats, which use a single hat in parallel, it assigns roles to each participant e.g. one reviewer reviews from the perspective of a tester, another as a developer, and another as the maintenance engineer. In my opinion, we can combine the thinking-hats with PBR as the thinking hats provide direction, while the role (e.g. as a developer) acts as a filter.

Conclusions

During this short tutorial you have had the opportunity to experiment with various 'thinking-hats'. I hope you have found the ideas useful and that you will find further opportunities to apply some of these ideas on your projects. Please let me know how you get on!

The final task

Please review this tutorial by using the six-thinking-hats. Your feedback will help to improve the material ☺

References and further reading

Link to Edward De Bono's site on the Six Thinking Hats
http://www.edwarddebono.com/Default.php

Details of Edward De Bono's revised and updated book on the subject
Six Thinking Hats, ISBN 0-316-17821-4

Link to my brief article on the six-testing hats
http://www.logigear.com/newsletter/six_hats_for_software_testing.asp

Link to Bret Pettichord's work on the original four schools of software testing
http://www.io.com/%7Ewazmo/papers/four_schools.pdf

Link to an online course on risk-based testing, from Cem Kaner
http://www.testingeducation.org/BBST/BBSTRisk-BasedTesting.html

Link to an article on how to apply risk-based testing, by James Bach
http://www.satisfice.com/articles/hrbt.pdf

Link to information on exploratory testing, by James Bach
http://www.satisfice.com/articles/et-article.pdf

Link to a comprehensive paper on the benefits of perspective based reading
http://iit-iti.nrc-cnrc.gc.ca/iit-publications-iti/docs/NRC-43603.pdf

Link to Alan Richardson's ideas on how descriptive words help us to test better
http://www.compendiumdev.co.uk/essays/descriptivewords/words.php

Acknowledgments

A number of people have helped me to formulate my ideas, review the content, etc. They include: Isabel Evans, Andrew Goslin, Mike Kelly, Jens Pas, Stuart Reid, Joanne Webb, and other friends and colleagues.