Test and Decision Theory

Bogdan Bereza-Jarociński
Enea Data AB
Box 232
S-183 23 Täby, Sweden
www.enea.se
bogb@enea.se
Phone +46-50-709-714 293

Motto:

"Of course, faith is a risk – but one I would never risk living without"

(Archbishop Desmond M. Tutu)

Abstract

In industrial reality, risk is typically assessed by estimating probabilities and consequences. This approach provides only raw *data*, but does not offer any *strategy* for decision making.

Software testing practitioners are commonly confronted with two decision-making situations: *what* to test and *whether to continue* testing.

Decisions like these are made – because of the lack of known strategies – on the basis of the experience, intuitive assessments and heuristic rules. The results are sometimes strikingly *wrong*.

A branch of statistics called *decision theory* provides a number of concepts and methods that can be used to make better decisions in a structured and controlled way.

Specifically, a method called *Bayesian Belief Nets* (BBN) has been developed for making decisions in complex situations. Tool support is available, which makes this method applicable in industry.

Contents

1. Introduction

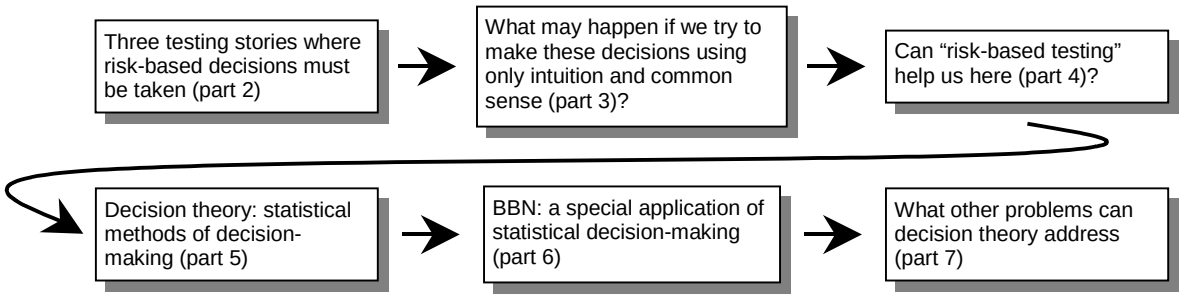
The way important decisions are made is sometimes shockingly primitive. The information may be very high quality: metrics, statistics, trend estimates and economic analysis. However, the decision itself, the very moment when someone says "we choose A rather then B or C", is mostly taken on the basis of experience, intuitive assessments and heuristic rules.

It may be so that A promises higher gain than B and C, even though the probability of failure is higher. Or perhaps it is the other way round - A does not promise as much gain as B or C, but it is much safer, i.e. the probability of failure is lower and the loss in case of failure much less. Who knows? Who will anyway remember this in half a year's time?

You may be ISO-9000 certified, CMM level 5, or TickIT-certified. All your processes may have names and descriptions, and they can be self-improving. However, there is one process that you are almost surely lacking: *the process of decision-making*. No, the decision-making is still mostly done in the fine honourable old way: closing your eyes, thinking hard and following your experience, your gut feeling, your emotional intelligence or your business instinct. This is done this way even if a given decision means comparing hundreds of inter-dependent variables and – provided you really tried to take them all into account - the only gut feeling you could expect was that of severe indigestion.

Now that I have grumbled long enough, it is time to have a look at the paper's contents.

Figure 1. The overall structure of this paper



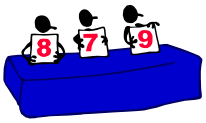
Just one more point before you start reading: the paragraphs that look like this one ("Century Schoolbook" font, greyish background and thin frame around) contain *examples*. Some are real, some are modified and some are fully fictitious. Their purpose is to illustrate the discussed subjects with the examples from tester's life.

2. An Appetiser: Three Short Stories to Lure the Reader

Terms like "statistical decision theory" or - even more – "Bayesian Belief Nets" may sound abstract, theoretical and scary, but actually they apply to very common, very simple everyday situations.

2.1 The Story about Test Planning

Classical risk analysis consists of three steps:



1. Identifying risks
2. Assigning *probabilities* and *consequences* to each identified risk
3. Calculating (probability * consequence) the numbers that reflect the relative *importance*, or urgency, of each risk.

Additionally, the analysis of available means to mitigate the risks (decrease their probability, or consequences, or both) can be performed.

It has been suggested [Ref. 7], that for the test planning purposes, the third variable (besides probability and consequence), called *testability*, shall be added. The value of this variable reflects whether a given risk can efficiently be addressed by testing.

Let us now study a fictitious example of risk analysis for a project.

Table 1. Results of the risk and testability analysis

Risk	Probability[1)]	Consequence	Importance	Testability
Risk 1	1	3	3	2
Risk 2	3	2	6	1
Risk 3	1	2	2	3

If the available time allows doing only one of these test activities, which shall be chosen?

Should we do the test addressing “Risk 3”, because it is sure to produce some results, even if they are not so important? Or should we rather do the test addressing “Risk 2”, because this is the most important. But it has low testability, i.e. test efforts may be wasted... Or maybe we should run 30% of the test cases for each category?

The risk analysis itself does not provide any answers, or the tools to make the “right” decision. What the right, i.e. optimum under the circumstances, decision would be, depends on the project goals and their priorities, and on the adopted risk strategies (minimise the possible loss or maximise the possible gain, for example).

2.2 The Story about Choosing Test Methods

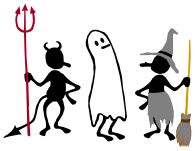
A project has chosen to design its test cases from a state-machine behavioural (black-box) model of the system under test. Two questions need to be answered:

1. What functional test coverage to choose: each transition at least once, each pair of transitions at least once, or more?

[1] All variables are estimated on a scale 1 – 3.

Importance = probability * consequence

2. For each tested transition, what inputs to test? All possible, all possible that cause the transition, one for each of the three equivalence classes (class 1: inputs that cause the transition; class 2: inputs that cause other transitions; class 3: ignored inputs), or just any one input that causes the tested transition?



To make this decision knowing what risks are taken, we need the data on how many more test cases each approach would generate and how much more time their preparation and execution would take. Besides, the probabilities and consequences of missing some faults if less coverage is chosen should be estimated.

Finally, when this data is available, a decision is to be taken. To make it, we need again to know the overall priorities (how important is quality? how important is time-to-market?) and the level of risks that is considered acceptable.

2.3 The Story of "Are-We-Ready-Decision"

First, we have taken take risks deciding how to prioritise our testing effort, what areas to test most [2.1]. Secondly, we have taken risks by deciding the levels of coverage, choosing some test cases and rejecting others [2.2].

Now we have been testing for two months and only three known faults are left. Using "fault trending" we estimate that there may be some 5 – 10 bugs unknown to us still in the code.

Shall we deliver? Again, we need a strategy on how to make this decision.

3. The Psychology of Decision-Making

In this chapter we describe a number of known facts about the way people act in dynamic, complex and probabilistic situations, like the situations described in the previous chapter. The described facts are not just hypotheses, but they have a solid experimental backing (the referenced positions contain more information) and are statistically significant. This *is* the way people make decisions, estimate probabilities and choose strategies in real life.

One must not forget, however, that the presented findings are only *statistically* significant, which means that, though most people act like that, considerable individual variations may exist.

3.1 Intransitivity of Preferences

Preferences are *transitive* when the following applies:

$$\texttt{if (result 1 > result 2) AND (result 2 > result 3)} \Rightarrow \texttt{result 1 > result 3}$$

In ordinary language, if Ann prefers Andrew to Bill and Bill to David, then – if her preferences are transitive – she is expected to prefer Andrew to David. Otherwise, Ann's preferences are intransitive (she likes Andrew better than Bill and likes Bill better than David, but given the choice between Andrew and David she prefers – surprise, surprise – David).

Psychological research has shown [Ref. 13] that people's preferences often are intransitive. When the result of a preference function depends on more than one variable, intransitivity may occur.

1. A test manager was asked at an internal development meeting whether her test team could help with some debugging activities, which required a good knowledge of the measurement instruments that her testers used. "No" – came the answer – "we must have time to achieve 95% statement coverage. This is our prime responsibility and more important than helping to find and correct bugs".

2. The same manager went then to another meeting and declared that locating and correcting known faults was more important than keeping the delivery date. "It is better to deliver a bug-free product two days late than to deliver tomorrow with known bugs and make a bad impression on the customer" – she claimed.

3. At a third meeting with project sponsors the test manager declared that keeping the delivery date was more important than achieving 95% statement coverage.

Her declared preferences were intransitive: (coverage > debugging) and (debugging > delivery) but (coverage < delivery). If her preferences were transitive, we would expect (coverage > delivery).

The reason can be that the motive behind her preferences was to make a "professional" impression. However, the contents of this vague term are slightly altered in different contexts. In the first two situations, technical considerations predominate. However, in the third context, the wish to appear "businesslike" becomes more important.

3.2 Time Preference and Delayed Gratification

The so-called *subjective utility* of a given gain depends on at least two variables: the value of the gain and the delay after which it will be received. Research [Ref. 10] has shown that there is a strong negative correlation between the delay and the subjective utility.

In other words, a thousand dollars to be received tomorrow is "worth more" than a thousand dollars to be earned in a week.

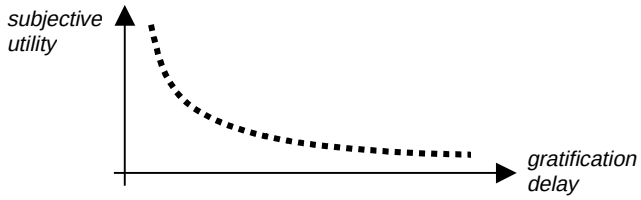


Figure 2. The negative correlation between subjective utility and gratification delay (with constant "monetary" value of the gain)

The strength of this tendency is negatively correlated to age and to some personality traits. Children generally have considerably lower tolerance to delayed gratification than grown-ups have. Some people have higher tolerance to delayed gratification.

In some respects is testing itself - as opposed to development - an activity with delayed gratification: development offers immediate "constructive joy", whereas testing and QA promises satisfaction later, when high quality has been achieved.

Our preference for immediate gratification may cause us to execute first the easy test cases or those test cases that find many – low-priority – bugs, instead of executing the most important test cases first.

Part of the lure of capture-replay tools is the capture functionality, which offers more immediate gratification than scripting techniques.

One of the principles for the successful introduction of test automation: achieve early success and advertise it, aims specifically at the managerial need for early gratification.

Introducing test automation without proper previous planning of the testware architecture is an example of how less value *now* may have higher subjective utility than more value - but later.

3.3 The Perception of Probability

Among many discoveries on the human perception of probability, two findings are especially relevant to the activities and decisions common in software testing.

Independent Events

People have difficulties grasping the idea of *independent random events*. This applies even to people with formal education in statistics and in probability calculus, whenever they control their activities intuitively. For example in roulette, getting a given number, say "21" for the third consecutive time seems to most players – judged from the way they do their betting - much less probable than getting the same number for the first time. Actually, it is the same, see figure.

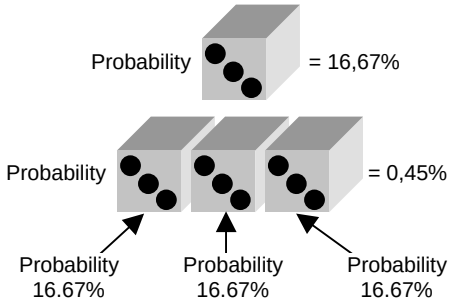


Figure 3. The probability of independent events

The question, whether software faults are independent or dependent events, is central for regression test planning. If they are independent, the same regression suite may be executed regardless of what faults have been found. If they are dependent, areas "around" already found faults must be tested more.

Fault tree analysis is the technique used to compensate for the human tendency to underrate the probability of the last event in long chains of independent events.

Making predictions about the number of the *left* bugs on the basis of the number and frequency of the *found* bugs, is only correct if the bugs still left are related to those

already found. But if they are "independent bugs", e.g. in a totally untested area of functionality or in a different part of the code, making such predictions is wrong.

Subjective Probability

The probability of the occurrence of the extreme values seems to most people less than of the probability of the "middle values" form an interval. For example, the experiment participants are instructed to place bets on different integers from the interval 1 – 100 and informed that the probability of each integer is the same (1%). Bets concentrate around the middle of the interval in spite of the fact that the probability of - for example – "50" is equal to the probability of "1" or "100" [Ref. 8, 9].

Figure 4. Subjective probability: low versus high probability

Similar misjudgements apply to other extreme values and to regular patterns. For example, there is a visible tendency to underestimate very high probabilities and overestimate very low probabilities. A regular sequence like 1-2-3-4-5 is often seen as less probable than an irregular sequence like 3-47-32-12-86.

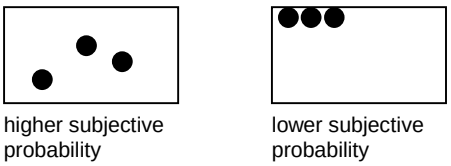


Figure 5. Subjective probability: regular versus irregular

Boundary value analysis is an effective test method because it looks for the faults caused by the developers who made the error of thinking that extreme values are somehow less probable/important than the values around the mean.

Generally, our intuitive estimations of probability, e.g. of the risks during development, of the risks during operation, or of the probability distributions of user activities, can be strikingly wrong. Therefore, techniques like fault tree analysis and the modelling of user activities should be used to get more reliable probability estimates.

Traditional risk analysis uses intuitive risk estimates by the participants, which is its primary weakness.

3.4 Ignoring the Size of Sample

People often make generalisations based on very small samples. The (statistical) significance of such samples is very low. However, most people (even with formal education in statistics) ignore the size of a sample when estimating probabilities in everyday life.

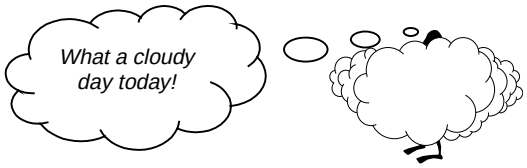


Figure 6. Ignoring the size of a sample when estimating probabilities

Ignoring the size of a sample supports "magical thinking", where inferences are drawn based on coincidences. For example, an accident happens to someone twice on Wednesday, which prompts this person to judge Wednesday as his or her "unlucky day". In reality, the probability of an accident happening twice on the same day of the week is quite high, 14%!

Assuming high product reliability on the basis of insufficient testing (too few test cases, too little "sample" of user actions) is the most blatant example of jumping to conclusions ignoring the sample size.

Another area where samples are typically very small, but conclusions are nevertheless formulated with high self-confidence, is process improvement. When two projects are different in some respect, and one of them happens to work better, the difference is immediately proclaimed as the *cause* (see as well "Causal Bias" [3.9]) of the "improvement". Specifically in the test area, this attitude causes rash and unreliable conclusion on the relative merits of test techniques, test tools and test automation.

3.5 Problem-solving in Dynamic, Complex Situations

In large, complex systems (an ecosystem, an economy, a society, an organisation, a large technical, computer or telecomm system) all elements are interdependent. Changing one element may have hard-to-predict consequences for the state of other elements and the system as a whole.

The decision-makers in such situations make a number of typical errors [Ref. 4].

Failure to Anticipate Side Effects and Long-term Effects

Faced with a complex problem with many complicated inter-dependencies, many decision-makers revert to a simplified model of how the system actually works. This creates the feeling of being in control and coping. For example, the decisions may take into account only the simplest, or the most urgent, or the most familiar problem, without regard to indirect and long-term outcomes.

Not testing in order to save money is a typical and well-known example of the failure to anticipate long-term effects.

Fault corrections that introduce new faults are examples of the failure to anticipate and take avoid side effects.

Long-time effects in software systems are vary hard to anticipate. Therefore stability testing and long-time tests are used, because these effects cannot be predicted in any other way.

The introduction of test automation to gain time in a project already late is an example of failing to anticipate side effects.

Using mainly young, inexperienced personnel for testing means sacrificing long-term considerations in order to solve an immediate problem (the lack of testing resources). In the long term, it will bring about harmful effects: low status of testing, low quality of testing, personnel turnover, etc.

Insufficient Monitoring whether Corrections Yield Required Results

Acting on a simplified, incorrect model as described in the previous paragraph, decision-makers often neglect the need to monitor the consequences of their decisions, and ignore apparent warning signals. The reality is not allowed to interfere with the comfort provided by using the simplified model.

This is called *focusing* in managerial lingo ;-) The decision-making should – besides choosing one of the options – provide a fallback plan and the means to monitor whether the expected results really materialise. This is often neglected.

Insufficient re-testing and regression testing are two examples of such behaviour (for software corrections).

Failing to gather process and product metrics is most widespread example of "insufficient monitoring".

Soothing Anxiety by Substitute Goals and Cynical Reactions

Not being able to cope with the complexity of the situation, decision-makers may revert to finding substitute goals or to concentrating on minor goals, which are less complex and provide the feeling of control. The feeling of being overwhelmed by the magnitude of the problem may result in cynical reactions: diminishing the negative consequences, playing down the lack of control as unavoidable, etc.

Two common examples of substitute goals in software testing are: to maximise the amount of submitted defect reports and to “play” with test automation tools.

Creating very detailed process descriptions or very formal quality systems that nobody wants to use, are other substitute goals in test and QA.

Generally, delivering the results for which there is no customer (test reports that management ignores, defect reports that developers disregard) means using substitute goals when the reality is too complex or too painful to bear.

Deliveries – internal or external – of untested code, making bitter jokes about the lack of configuration management or chaos with defect reporting, are all symptoms of “cynical reactions” when nothing is being done about the complexity problem.

3.6 Setting Goals

The difficulties of the decision-making situation may stem not only from the system complexity, but from the complexity of our goals as well. When the goals are multiple and interdependent, we have serious additional complexity to tackle. This situation may be made worse by our unsatisfactory representation of the goals. Goals that are non-specific, unclear and implicit make decision making virtually impossible.

In similar situations, decision-makers may resort to a number of different techniques to deal with the complexity. Hard *prioritisation* reduces the number of goals. *Delegation* removes the responsibility for given goals elsewhere. *Reactive behaviour* – tackling only immediate problems and short-term goals as they appear – soothes negative emotions and simulates control and activity [Ref. 11]. The candidates for substitute goals are those goals that result in "pleasurable", rewarding activities [Ref. 3], which of course need not be the most important from other points of view.

> Confused middle-level managers, recently lifted to their positions from developer roles, frequently find comfort in the "micro-management" of the technical details with which they feel comfortable.
>
> Products have many different quality attributes[2)], which often have negative coupling (i.e. more of one attribute causes less of another attribute). Quality goals seldom take this into account.
>
> Similarly, complex requirements (especially non-functional requirements) are often brutally "simplified", which more often than not means discreetly dropping some "troublesome" requirements.

3.7 Coping with Temporal Changes

Dynamic systems that change over time provide special difficulties for decision-makers. The most common is the extrapolation of expected future trends and states on the basis of the current situation. We have a tendency to think in linear terms and cope poorly with non-linear processes.

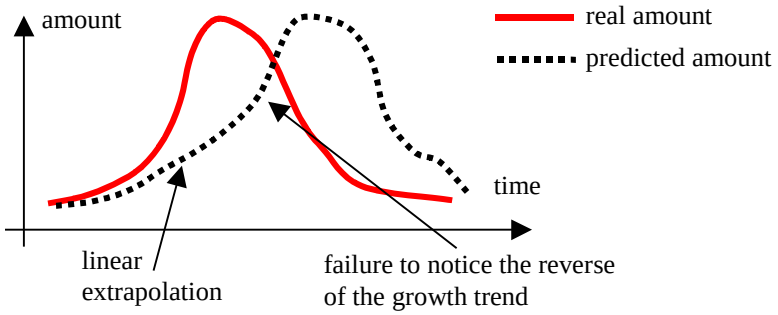


Figure 7. Prediction errors for a non-linear development

[2)] Many "lities", a "ncy" and a "ness": efficiency, functionality, maintainability, portability, reliability, usability, expandability, interoperability, reusability, integrity, survivability, correctness, verifiability, flexibility...

Delayed feedback is another confusing effect, very common in many systems. When the

outcomes of our manipulations occur after some delay, many people become completely confused and predict random behaviour (or make "magical" assumptions) for perfectly deterministic systems. This may result in *ritualistic behaviour*, for example repeating old actions in spite of their inefficiency.

Changes in the trends of the numbers of executed tests, the numbers of found and corrected faults etc. can be difficult to interpret. Premature decisions based on the facts interpreted without the regard to the "time skew" between stimuli and outcomes are common.

For example, many deliveries to the system test have too poor quality to allow testing. To address this problem, pre-delivery "smoke" testing is introduced in the development project. After some weeks, the deliveries still have poor quality. Pre-delivery testing is scrapped. Actually, after the period of initial difficulties, the pre-delivery testing was about to bring about the required results when it was cancelled.

A very similar situation occurs often with the introduction of test automation. When test automation fails to deliver miracles fast enough, it is stalled before it can start delivering results.

Two people from the testing group quit. The management fails to interpret it as a warning, believing it is a part of normal turnover pattern. Some time later, the discontent has reached such proportions that almost everybody from the group quits.

The number of faults found in consecutive internal deliveries falls down. This is used as the rationale for the delivery decision. Actually, it was only a temporary drop due to the fact that these deliveries contained very little new functionality.

Ritualistic behaviour occurs when there are attitudes "we have always worked this way" and the "not-invented-here" syndrome appears. Experienced project leaders and test managers often re-use the methods and patters that were successful in previous projects, without prior analysis.

3.8 Cognitive Dissonance

The *cognitive dissonance* occurs when our internal representation or expectations do not agree with the observed facts or actual outcomes. Typically, cognitive dissonance causes negative emotions, which we try to remove by removing the dissonance. To remove the dissonance, we can either change/adjust our internal representation and expectations, or re-interpret the observed facts so that they seem to agree with the expectations.

Cognitive dissonance causes many basic test-related problems. It is the reason why individuals and organisations should not test their own products. Cognitive dissonance explains why "not bugs found" is considered good news, and "a hundred bugs found" is bad news, even if it is the other way round from the point of view of the final product quality.

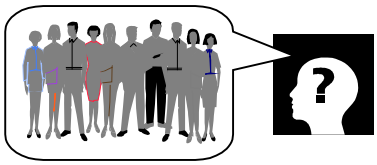
3.9 Causal Bias

People tend to assume the existence of a causal relationship where there is only the evidence of a positive correlation. For example, watching violent videos if often given blame for violent

teenage behaviour. What is more probable, there is a third variable, for example being brought up in a violent environment, that is the common cause of both violent behaviour and the interest in violent movies.

See section "Ignoring the Size of Sample" [3.4] for examples.

3.10 Group Thinking



The decisions made by large bodies – committees, review teams – are commonly thought of as better decisions. There are factors that may make collective decisions better, for example the access to more information and the fact that not everyone makes the same error. On the other hand, a phenomenon called *group thinking* may make the results of collective decision-making worse than individual decision-making.

Group thinking is the way that group members behave in order to get acceptance from other members and to enjoy the feeling of emotional safety that group membership yields.

Examples of group thinking may be inefficient reviews. Due to group pressure "let's finish this and go for lunch", or because less senior participants do not dare to voice their doubts, or because the responsibility is not assigned to any individual, documents that should have been rejected are accepted.

Another example of group thinking is when old test methods and the old design of test cases are never questioned.

Testers are sometimes openly encouraged to engage in group thinking, when they are requested to drop some defect reports or take the responsibility for debugging, or give a go-ahead to a delivery that they know is not good enough.

3.11 Conclusions

In a way, the findings described earlier in this chapter, though well known for professional psychologists, come as a surprise to many readers. It is shocking to find that people seem incapable of acting rightly in complex and dynamic situations. Our intuitive grasp of statistics and probability is mostly incorrect. To put it bluntly, our behaviour often seems stupid.

This "stupidity" is in fact a number of very adaptive traits that ensured the survival of our species. The ability to solve immediate problems fast and to cope immediately with approaching dangers required a number of filtering techniques that kept information processing to the minimum and allowed fast decision-making in situations with incomplete information.

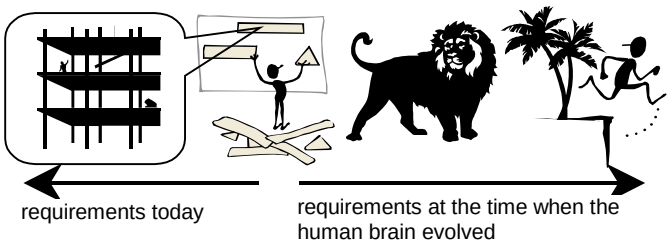


Figure 8. The ability to cope with long-term and short-term problems

On the other hand, many of the traits required in complex situations of today's technological civilisation, were useless or even dangerous in the time when the human brain initially evolved. Individuals, who had a tendency to engage in the analysis of the situation and the anticipation of side effects and long-time repercussions, were generally eaten by predators long before they were halfway through these demanding intellectual tasks...

To be able to cope more successfully with the challenges of living in a complex world and building complex, dynamic systems, we should recognise both our strengths and our deficiencies, and act accordingly. Human brain is unsurpassed in creative work and in making quick and reasonably good short-term decisions without engaging the heavy apparatus of more complex thinking. On the other hand, we are very poor in making good estimates and correct decisions in complex and dynamic situations. Therefore, we should turn to available tools for support. These tools – based on statistics – and their usage are described in the following chapters.

4. What is a Risk-Based Testing?

Using a search engine for "risk-based testing" gave 127 hits. It is a common term today, especially applied to testing e-commerce and other Internet applications.

Generally, *risk-based testing* means that testing is concentrated in areas where the risk is highest [Ref. 1 and 7]. The decisions what to test and when to stop testing are based on the analysis of the involved risks.

In some respect this approach resembles Statistical Usage Testing (SUT). SUT defines risk as the probability of a given "usage". Test cases are generated and prioritised according to this probability. However, the definition of risk used by most "risk-based test methods" is broader. It includes both risk *probability* and risk *consequences* (test more where the consequences of failure are greater). Besides, both *operational* and *developmental* risks are addressed (more testing where the risk of errors in development is greater).

Risk-based testing is about how to identify existing and to predict possible risks, about how to assess their probabilities and the consequences, and how to mitigate them (i.e. decrease the probability or mitigate the negative consequences) with the help of testing. The methods described for these activities are mainly heuristic and designed specifically for the usage in software and system testing.

This paper supplements risk-based testing in three respects.

1. Psychological obstacles to efficient heuristic methods. In chapter “The Psychology of Decision-Making” [3.] we discuss why – in spite of the participants’ experience, the usage of defined processes and in spite of performing meticulous inspections - heuristic methods may fail to produce correct decisions.
2. Some examples of the application of formal, statistical methods for the decision-making in test process are presented (“Statistics: Decision-Making Under Uncertainty”, [5.]).
3. One well-developed method with tools support for risk assessment and analysis is presented in "Making Decisions Using Bayesian Nets", [6.].

5. Statistics: Decision-Making Under Uncertainty

5.1 Two Types of Uncertainty

“Statistics is the science of decision making under uncertainty” [Ref. 2]. Actually, most decisions fall into this category, as (future) outcomes of a (present) decision are never completely sure.

There are two kinds of *uncertainty*. The first kind is uncertainty *due to randomness*. Casting dice we know that the probability distribution of possible outcomes is rectangular: 16,667% for each of the six possible outcomes. However, we do not know for sure what outcome will happen next – therefore uncertainty.

The second kind of uncertainty is when we do not know *which laws of randomness apply*, or - in statistical lingo - what is *the state of nature*. It is like using a dice that is perhaps false, we have no clue to what the probability distribution for the six outcomes is. This is the kind of uncertainty we commonly face in software testing situations. We do not normally know the probability distribution of the *user actions* and we do not know the probability distribution of the *faults*.

In order to tackle this kind of uncertainty, experiments can be made and observations taken. By casting a suspicious dice a few hundred times, we can get a fair knowledge of what its probability distribution actually is.

5.2 The Consequences of Uncertainty in Software Testing

Not knowing the probability distribution of user actions is an obstacle to estimating whether our test suite is *representative*, i.e. whether it covers all use cases that will occur during operation. Not knowing whether the test suite is representative, we cannot estimate the *significance* of the test results, i.e. the probability that test results faithfully mirror the results of the system in operation.

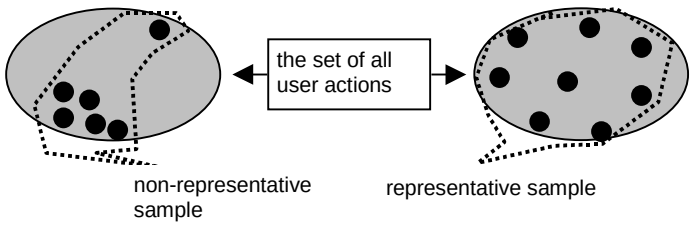


Figure 9. Representative versus non-representative sample

This uncertainty is successfully – at least in theory – addressed by the techniques like Statistical Usage Testing [Ref. 19] or Software Reliability Engineering [Ref. 12]. By gathering reliable information on the probability distribution of user actions (this is the main *practical* difficulty of these methods), we can generate representative samples of these actions, i.e. reliable test suites.

Still, the other uncertainty remains: the unknown probability distribution of the faults present in tested systems. There is a lot of anecdotal information available: "faults tend to appear in clusters", "there are on average X faults for Y lines of code", "faults appear on boundary conditions", "faults appear in complex modules", etc. However, not knowing the probability distribution, we cannot estimate whether it is likely that a given "rule" applies to the system we test now.

5.3 Utility

You gamble by casting a coin. There are three kinds of bets:

1. Heads: you win £500, tails: you lose £500.
2. Heads: you win £10, tails: you lose £10.
3. Heads: you win £10000, tails: you lose £2000.

Which bet is best for you? From the point of view of the win *value*, the bet number 3 is the obvious winner. However, value is not the same as *utility*. Utility is "a function on the set of prospects" [Ref. 2]. In this example, "prospects" are the (monetary) value of the win, the value of the loss and the win/loss ratio. The probabilities are constant, but in many other situations the probability of winning and the probability of loosing are "prospects", too. In other words, utility - the *dependent variable* - is a function of "prospects", which are the *independent variables*.

The shape of the *utility function* is different for different people, different organisations and different situations. For the author at the moment of writing, both loosing £500 and loosing £2000 feels like too much; therefore the bet number 2 has the highest usability. For someone who felt more at ease with the risk of loosing £2000, the bet number 3 would probably have the highest usability.

The first step in any decision-making process shall be defining what usability function applies to it. For example, you cannot make a good decision on the "release or continue testing"-dilemma unless you have identified all the "prospects" and have a relatively well-defined utility function on them.

The list of all the "prospects" can be long:

- the value of hitting the market window
- the cost of missing the market window (as the function of the time)
- the risk of failures in operation
- the cost (money loss) of failure in operation
- the probability of finding and correcting more faults (as the function of the time and resources)
- etc.

5.4 Some Decision Strategies

Minimising the Maximum Loss

This strategy defines utility as the function of the maximum value of the loss (and nothing else).

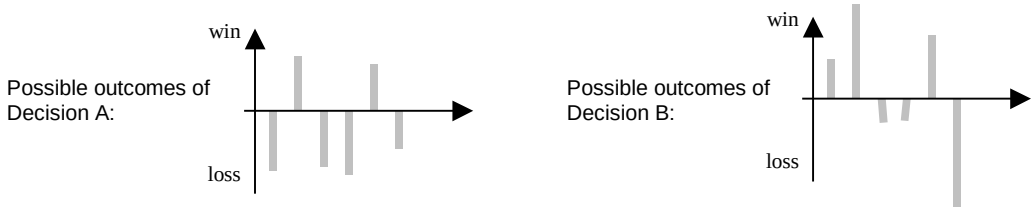


Figure 10. Lists of possible outcomes for two different decisions

Using the strategy of "minimising the maximum loss", Decision A will be chosen, even if Decision B looks better is some respects (higher wins, for example).

Minimising the Average Loss

Using this strategy, we would choose Decision B, as the *average* loss of all its possible outcomes is lower than for Decision A, in spite of that one of the outcomes is the loss largest of all.

Other Strategies

There are other strategies, including more complex ones. They are not discussed in this paper.

What If the State of Nature is Unknown?

What decision strategies can be applied when the "state of nature" (see "Two Types of Uncertainty" [5.1]) is unknown, i.e. when we do not know what probability distribution applies to a given area? After all, unless we have some clue to what may happen, we may as well choose our actions quite randomly?

This is not necessarily the case. Let us study the following example.

There are *two possible states of nature*: either there is a bug or there is no bug, but we do not know the probability of either state.

Assume that *three actions are possible*: we may (1) prepare not to test at all; (2) prepare to test but not to debug; (3) prepare to test and to debug.

Now let us assign the loss (loss is just a kind of "negative utility", see "Utility" [5.3]) to every combination of the state of nature and the chosen action.

	do not test	test and not debug	test and debug
there is no bug	0[3)]	1	3
there is a bug	5	2	0

Unless there is some indicator, which is correlated to the existence of bugs, the choice of actions can be random.

We consider using cyclomatic complexity for each module as the indicator of the fault existence. Let us presume the following correlation have been observed:

- high complexity: 60% of the modules contain a bug
- middle complexity: 30% of modules contain a bug
- low complexity: 15% contain a bug

Now can all possible strategies [4)] be evaluated: The *expected loss of utility* (or simply "loss") can be calculated for each. Here, it will be presented for just four of them:

	never test	always test and debug	low: do not test middle: test high: test & debug	low & middle: do not test high: test & debug
if there is no bug	0	3	1.9[5)]	1.2
if there is a bug	5	0	1.35	2.25

Which strategy is the best? Impossible to say, unless we make assumptions on the overall bug probability. For example, if the overall bug probability is 90%, the strategy "never test" would yield an *average loss* 90%*5 + 10%*0 = 4.5. The strategy "always test and debug" would yield, not surprisingly, a much lower average loss 90%*0 + 10%*3 = 0.3. But, if the bug probability was only 20%, the average loss of "never test" (20%*5 + 80%*0 = 1) would be less than the average loss of "always test and debug" (20%*0 + 80%*3 = 2.4)![6)]

[3)] Assigning the values of loss is difficult, and it must mirror our overall testing strategy. In this example, "loss" is defined only from the point of view of the product quality enhancement. Therefore, to "test when there is no bug" is just a wasted effort. Naturally, testing when there is no bug provides very valuable information, but this fact is ignored here to keep the example as simple as possible.

[4)] There are 27 of them in this example: the number of all possible three-element (three different levels of complexity are used) variations with repetitions of three elements (there are three possible actions) equals $3^3 = 27$.

[5)] 0 * 85% + 1 * 70% + 3 * 40% = 1,9; all other calculations follow the same rule.

[6)] All right, then, maybe the loss values assigned in this example to different outcomes should after all be revised again... ;-)

The so-called ***Bayes***[7] ***strategy*** is the strategy that minimises the average loss (as defined in the calculations above) for a given assumed set of probabilities. In other words, using Bayes strategy, we can find "the best" strategy (in a Bayesian sense, i.e. as defined above) for any assumed set of probabilities. Even if the state of nature is unknown, we can make a number of "what if"-analysis to support our decision-making process.

6. Making Decisions Using Bayesian Nets

The end of the previous chapter does not look like anything you could with a clear conscience recommend to use at, or even before a management meeting. The calculations are tricky, even though only three simple variables are involved. In managerial practice, we would like to be able to handle the situations where there exist *many interdependent variables*. Besides, it would be desirable if calculations could easily be re-done whenever new data becomes available.

This is actually what BBN - Bayesian Belief Nets - have to offer [Ref. 5 and 6]. Commercial tools that support them is available as well [Ref. 16].

6.1 What is a BBN?

BBN is a graph representing relationships among variables. Nodes of the graph represent variables, and the links are relationships (causal or relevance) between pairs of variables. In this respect, BBN-graph is very much like the graph for cause-effect analysis or for fault-tree analysis.

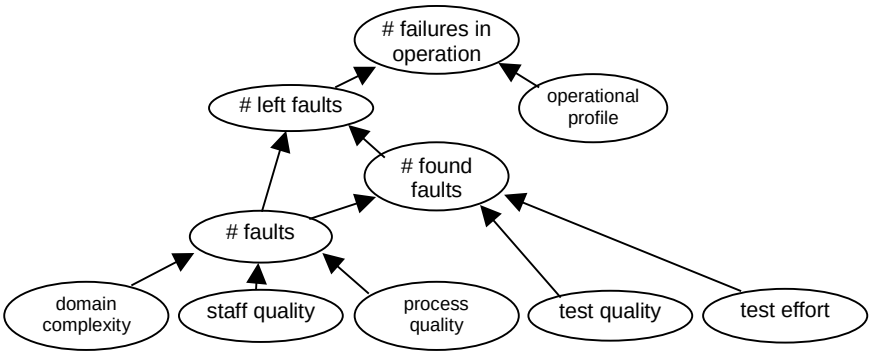


Figure 11. An example BBN graph

BBN graphs can be used in both directions with data of different quality and on different scales (at least ordinal[8]). For each node, there are associated probability tables. For example, if "staff quality" is measured on a simple ordinal scale "low", "middle" and "high", associated probabilities can be defined as - for example - 10% "high", 60% "middle" and 30% "low". Provided the value of a given variable is well known, then the probability 100% can be assigned.

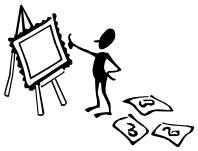
[7] T. Bayes was an 18th century English mathematician.

[8] Values on the *ordinal scale* are ordered, but no arithmetic operations are possible with them.

For the links, the associated values denote how much the variance of the value of the dependent variable (the node at the "end" of the link) is explained by the value of the independent variable (the node at the "beginning" of the link).

If the probability distributions of the independent variables are not known, then the inferred values of the dependent variable will have a rectangular probability distribution. For example, if (see *Figure 11. An example BBN graph*) "#found faults" and "operational profile" are not known, then the resultant "#failures in operation" will have the same probability assigned to all values (or value intervals). Having "the same probability assigned to all outcomes" means that nothing can be predicted about which outcomes are more probable than other outcomes.

6.2 BBN Usage



BBN graphs can be used in two ways: for *prediction* and for *updating previous "beliefs"* (expert opinions on the probability).

In the example graph above, we could - by making all assumptions - predict the "#failures in operation".

When, during testing, the "# found faults" becomes known, it can be entered directly into the BBN and all other values will automatically be updated.

Entering the actual "#failures in operation" (when it is known) will work all the way down the graph, updating the values of the lower nodes! Provided our estimates of how strongly variables depend on each other are correct, learning about the high number of failures in operation will automatically update our earlier estimates of, among other, "test quality" and "test effort"!

7. Decision Theory: the Missing Link?

Three elements are present in every testing situation: *management*, *test techniques* and *domain knowledge*.

1. Management organises test activities and secures the connection to other development activities and to the line organisation.
2. Test techniques are about how to choose and design test cases.
3. Domain knowledge is necessary to know what is most important to test from the user's point of view and to generate the expected outcomes for the test cases.

The connection among these elements is often chaotic.

> Transaction flow testing is chosen for a test project of a banking system. This decision is made without full control of the involved risks, as it would require domain knowledge.
>
> The test manager plans test activities, resources, lab equipment etc. without regard to what level of risks her decisions entail. To achieve the required confidence in the quality of the delivered software, the amount of testing and resources must be analysed taking into the account the chosen test techniques.

Perhaps, the use of formalised decision-making methods can provide the missing link.

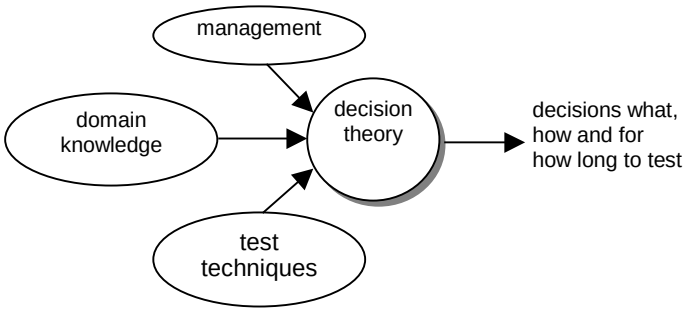


Figure 12. Using decision theory to bind together elements of the test process

8. Sources and References

8.1 References

1. Amland, S. *Risk-Based Testing*. Paper at "EuroSTAR" 1999. sa@avenir.no
2. Chernoff H., Moses L. E. *Elementary Decision Theory*, 1959
3. Csikszentmihalyi, M. Flow: The Psychology of Optimal Experience, 1991
4. Dörner D. *The Logic of Failure* ("Logik des Misslingens"), 1997
5. Fenton N., Neil M. *Making Decisions: Using Beyesian Net and MCDA*, 2000
6. Fenton N., Neil M. *Bayesian Belief Nets.* "Software Testing and Quality Engineering", 2000
7. Gerrard, P. *Risk-Based Testing*, 2000. Systeme Evolutif Limited, www.evolutif.co.uk/presentations/RBTpresentation.pdf
8. Kahneman D., Tversky A. *Subjective Probability: a Judgement of Representativeness.* "Cognitive Psychology", 1972
9. Kozielecki J. *Behavioural Decision Theory* ("Psychologiczna Teoria Decyzji"), 1975
10. Lee W. *Decision Theory and Human Behaviour*, 1971
11. Lindblom C. E. *The Science of 'Muddling Through'*. "Readings in Managerial Psychology", 1964
12. Musa J. D., Muda J. D. *Software Reliability Engineering: More Reliable Software, Faster Development and Testing*, 1998
13. Tversky A. *Elimination by Aspects: a Theory of Choice*. "Psychological Review" 1972

8.2 Other Sources

14. Agena Ltd., http://www.agena.co.uk ("*Agena Ltd is a consultancy specialising in risk management and decision support for business-critical and safety-critical computer-based systems"*)
15. Fenton N., Pfleeger S.L. *Software Metrics: A Rigorous and Practical Approach*, 1998
16. Hugin Expert A/S, http://www.hugin.com ("*the premier provider of Bayesian Network reasoning technology*")

17. Links on: *Decision Support and Bayesian Belief Networks, Software Engineering, Metrics and Reliability*, http://www.agena.co.uk/links.html
18. Norman Fenton's home page: http://www.csr.city.ac.uk/people/norman.fenton
19. Statistical Usage Testing: http://www.q-labs.com/port_sut.html