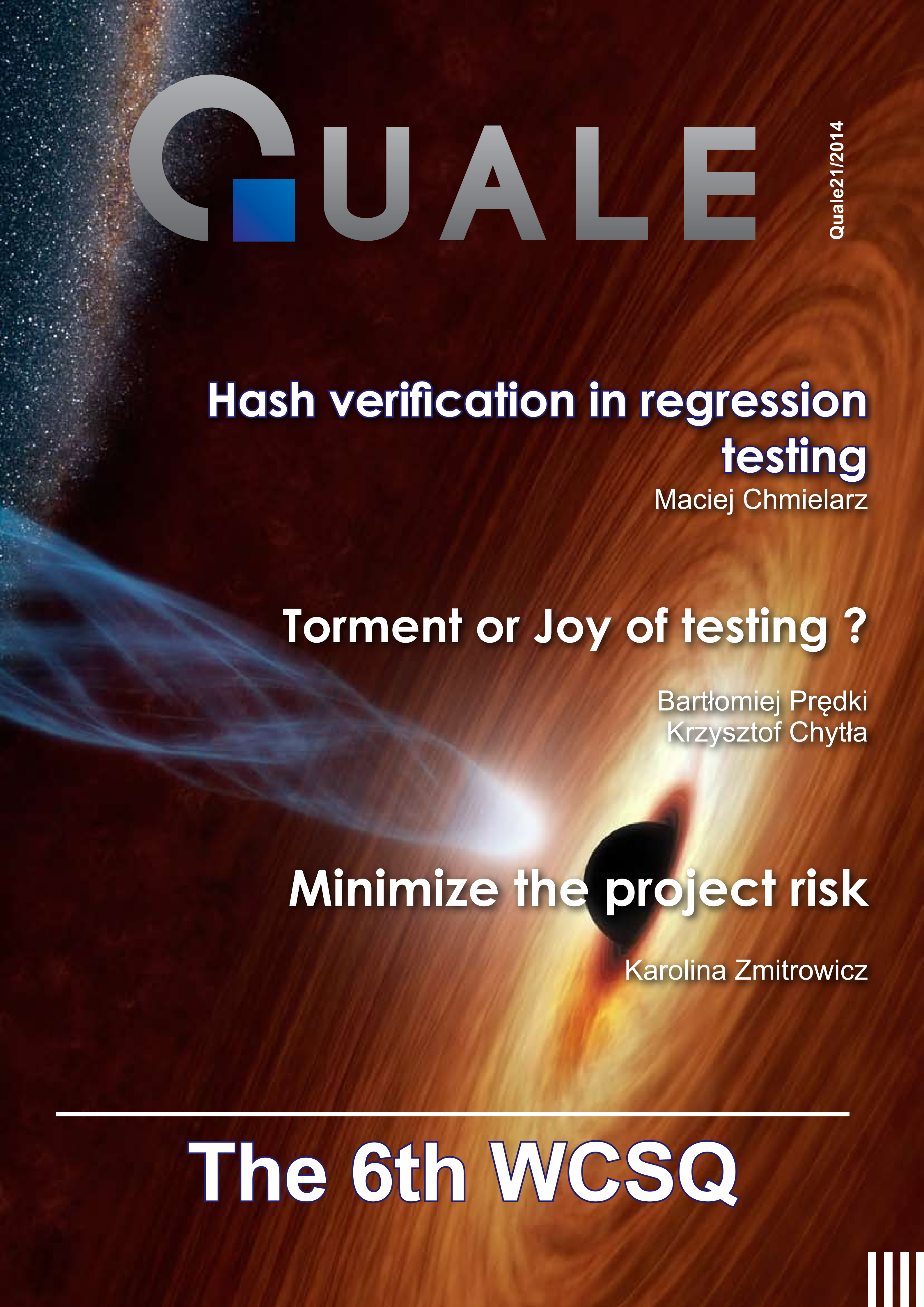


QUALE

Quale21/2014

Hash verification in regression testing

Maciej Chmielarz

Torment or Joy of testing ?

Bartłomiej Prędki
Krzysztof Chytła

Minimize the project risk

Karolina Zmitrowicz

The 6th WCSQ

Content

EDITORSHIP

Chief editor
Karolina Zmitrowicz
karolina.zmitrowicz@quale.pl

Deputy chief editor
Krzysztof Chytla
krzysztof.chytla@quale.pl

Editors
Bartłomiej Prędki
bartlomiej.predki@quale.pl
Michał Figarski
michal.figarski@quale.pl
Piotr Poznański
piotr.poznanski@quale.pl

Cooperation:
Tomasz Olszewski

Website:
www.quale.pl/en (ENG)
www.quale.pl (PL)

Facebook:
http://www.facebook.com/qualemagazine

Krzysztof Chytła

Torment of testing

Greetings my respectable fellow Editor! How is life? I hope everything's fine or at least better than on my end of the wire. I've been suffering from a spring depression or mild seasonal affective disorder, if you like. Bad weather really brings me down and causes that gloomy, melancholic mood. Surprisingly, there is at least one positive downside of such state of mind – it makes you wonder. I thought that some of my reflections on Weltschmerz are worth sharing with you. I hope you will spare a minute to read or even support your old buddy with kind word. Let's get started.

Existential "to be, or not to be" often haunts the minds of testers but how can it be that life is so miserable? How can I go on, from day to day [Freddie Mercury], stumbling upon lines and branches of code tangled into knots of mutually dependent functionalities? Mechanically repeating manual test cases like a maniac, over and over, from 9 to 5 – often longer – five days a week… and all for nothing. Yeah, nothing. Those once pesticide prone buggies evolved, lurk hidden deep inside, invisible even for a well trained eye. I haven't filed a single defect report in weeks. Kingdom for a bug! [Shakespeare] Truly depressing. I guess I need to consult my shrink before I drown in the pool of apathy and sadness. Is it sane to feast upon someone else's failures, to laugh when it doesn't work?

Whose fault is it when it doesn't work? Tester's – it's so obvious! Project delays? Blame the tester. We've been through that list last time. It is so hard to explain that soft had already been broken when you first touched it. In such cases you can almost hear you brain singing "It wasn't me" along with Shaggy. "It's not a bug, it's a feature" is as good as it can get however "stack trace or GTFO" amplified with a nasty green is much more common. As a tester you need to be cool as a cucumber. Restrain yourself from slapping faces and breaking office stuff. Keep calm and carry on testing like a pro! [GB].

I know, I know. All in all, everything is fine and software really works as expected! Things which don't aren't real issues or bugs but unrealistic scenarios and vivid fantasies. "Those aren't the droids you're looking for" [Star Wars] young testing padawan blinded by the Force of quality.

Thank you? "Thank you" is not on the vocabulary of most people facing testers. How come that testers who do nothing but point out errors need to be thanked? That's preposterous.

On the other hand releasing software on Fridays past 5 PM is not. Thou shall test till your eyes bleed as playing the low effectiveness of late night testing card doesn't break through to the minds of managers. Instead one should be proactive and volunteer for the Saturday crunching just in case and hope to get paid for the overtime. Remember, depression has nothing to do with lack of ambition.

Automation brings testing to a whole new level? Develops processional competence? Oh no, dear Editor Sir, by no means will I help Skynet [Terminator] gain control over the human kind!

My fragile dreams have been broken [Anathema]. All what's left is torment of testing.

Huh. That sounds bitter but I'm better off having thrown it all out. Actually I feel a bit - or even a byte - better. Sharing thoughts, even those melancholic ones, definitely helps you clear you mind. It's been really nice writing to you however there's something else I should have done a long time ago. No more time to waste – it's high time for holidays!

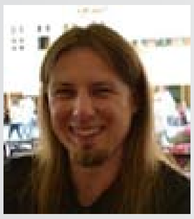
AUTHOR

Krzysztof Chytla

Test manager, designer and automation specialist with wealth of experience in embedded systems domain. Participated in big international projects assuring the highest product quality. Flesh and blood tester curiously analyzing rapidly expanding world of new technologies.

Author of translations and publications. Wroclaw University of Technology, Faculty of Electronics graduate. Trainer and coach passionate about acquiring and sharing knowledge.

On a personal note big fan of fantasy, science fiction and board games accopanied by a a glass of single malt whisky - an editor's best friend.



Bartłomiej Prędki

Joy of testing

Actually, I had wanted to start with "Let me welcome you, dear Deputy Editor in Chief" but I noticed that the official approach to greeting brings you depression and anxiety. Therefore I'll make it less formal – Hi there, my fellow! I do not know where your grief and sadness come from – just a few days of bad weather shouldn't be a surprise in our climate zone. When it rains, things start growing here and there - so don't worry, just be happy like Bobby McFerrin plus hope there's no ground frost or you might slip and get into real trouble. And – by the way – to call yourself old, you would need to spend much more time on wondering.

You write about anguish of repetitive tasks done throughout countless number of hours and days. As they say - that's what the job is like. I also don't understand your complaints on manual testing – you definitely need to change your approach! Your testing should be started with words: "give me your kings, let me squeeze them in my hands!" [Freddie Mercury]. I could have also written that if the bug still remains, change pesticides – but nothing like that will take place. Sometimes, when there's a lot of manual routine tasks to perform I add a bit of exploration to it. How it's done? Very simply – after a few planned test cases - BOOM! – I unexpectedly jump to another area. This is because the enemy does not expect an attack from the left flank. The only person depressed at such moment would be the author of the code. Sweet mercy is nobility's true badge [William Shakespeare].

Situations called "it was already broken" or "it's not a bug, it's a feature" don't cause any stress for me at all. Complying with professional approach, I ask for an official change request when it was already broken (e.g. defect number from the previous release) or for a chapter in the documentation describing alleged "feature". And it's not like a relaxing massage; it's just a regular slap in da face. And of course it's done with your hand, hence manual! Plus it really works! A separate issue is that professional approach should not be applied to individuals just occasionally and merely benefiting from the shower...

Finally it works? What a surprise! And they don't say "thank you" to testers because we just point out errors? Oh dear, I'm depressed and torn and… Oops! I just dropped the s**t I was about to give. I'll tell you what to do, young software padawan blinded by the Force of quality. Shortly after having found several critical bugs - just pass by the area occupied by C++ Dark Forces, hoarsely whispering "I am your father" [Star Wars]. Stampede guaranteed...

I won't tell you what I would do to people planning releases just before the weekend (did I mention something about the censorship last time?). But as an absolutely non-vindictive person, I let myself to bounce-back during the weekend. As you know, only testers and developers work then, ergo, this means no control. The natural thing is that time flies by faster while having a bit of warming liquids. Truly I say unto you - agile testing takes on a whole new meaning then. And you can go home in the morning and sing: "you shook me all night long" [AC/DC].

I don't even want to start the automation topic discussion. It's kind of "blondes vs. brunettes" or "petrol vs. diesel" dispute. As I mentioned, manual slap from the tester hurts much more than a virtual ping from the automation software.

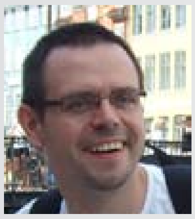
You say torment, huh? Sure, a little bit, like everywhere else. Joy and fun? It's nowhere to be found but here, where testers are knights and bugs are the dragons.

Still, I'm glad that you feel better now. Sometimes you need a different view to see that the situation is not always as bad as it looks.

Hey wait, what holidays? At this time of the year? How could you even dare to complain ...

AUTHOR

Bartłomiej Prędki



I've started my professional experience in 2004 as a tester of mass-market mobile applications. Within next years I gained an experience in Testing and Quality Assurance areas, mostly focused on Telecommunications industry.

During my career I was involved in testing, managing testing processes, training, technical support, requirement analysis, recruitment, technical documentation creation and review.
Besides my mobile and telecommunications experience, I was also involved in financial and banking systems related projects. Currently I possess the role of QA Team Lead.

I'm a holder of two ISTQB Advanced certificates: Technical Test Analyst and Test Manager

I live and work in Wroclaw, Poland.

Karolina Zmitrowicz

Minimize the project risk

Build good business requirements

ABSTRACT

What is this article about? Not much about the typical reasons of projects failures, as there are plenty of papers on this topic. We will focus on one of the core problems with IT projects – requirements. Moreover, we are going to focus on business requirements, their meaning and impact on projects. I believe we all know that business requirements are of crucial impact for any project, as they create a base for project planning, estimations, scope and content definition and realization of works.

The role of requirements in IT projects is not the only subject of this article. It is important to know the meaning of requirements, but it is even more important to know how to build requirements in a way allowing to avoid typical risks and problems. Therefore we will talk about principles of building good requirements.

What is the problem?

To be able to define a solution for a problem, we need to identify the problem first. In this case, we are talking about reasons why projects fail. Let's then consider why do they fail. We can start from analyzing common statistics and researches, for example, the Chaos report. The report presents a set of statistics and their interpretation prepared by Standish Group. Among others, this report shows the main reasons of project failure, most important success factors and other statistics related to the realization of IT projects. Let's have a look at some statistics showing the main project success criteria (Table 1 Project success criteria [1]).

CHAOS REPORT

1994	1999	2001	2004	2010, 2012
1. User Involvement 2. Executive Management Support 3. Clear Statement Of Requirements 4. Proper Planning 5. Realistic Expectations 6. Smaller Project Milestones 7. Competent Staff 8. Ownership 9. Clear Vision And Objectives 10. Hard-Working, Focused Staff	1. User Involvement 2. Executive Management Support 3. Smaller Project Milestones 4. Competent Staff 5. Ownership	1. Executive Management Support 2. User Involvement 3. Competent Staff 4. Smaller Project Milestones 5. Clear Vision And Objectives	1. User Involvement 2. Executive Management Support 3. Smaller Project Milestones 4. Hard-Working, Focused Staff 5. Clear Vision And Objectives	1. Executive Support 2. User Involvement 3. Clear Business Objectives 4. Emotional Maturity 5. Optimizing Scope 6. Agile Process 7. Project Management Expertise 8. Skilled Resources 9. Execution 10. Tools & Infrastructure

Tab. 1. *Project success criteria*

As we can see the most important success factors can be defined as follows:

- Executive support
- User involvement
- Clear business objectives
- Emotional maturity
- Optimizing scope

Smaller project milestones are also of great importance, as well as clear statement of requirements. How does it deal with requirements? Let's check one more thing before answering this question.

Success factors are interesting, but reasons of failure should be as interesting as well. Summarizing information coming from different researches, we can say, that the primary reasons for failure are:

- Lack of user involvement. Poor user involvement results in the fact that the solutions we create may not meet user requirements or do not support users tasks. In the end, the user receives a product that does not meet expectations and is not useful in real-life usage.

- Lack of management commitment. This is serious problem, as if you don't have management commitment, there is a risk that there is nobody responsible for the outcome of a project. And if no one is responsible – no one cares about its success or failure…

- Problems with requirements and specifications. Issues with requirements and specifications is a very broad topic covering all known problems with requirements – bad quality of requirements, incomplete requirements, requirements that do not meet specific acceptance criteria or cannot be measured and tested. All these problems cause that the product being developed is based on wrong or partially wrong assumptions, therefore the risk of producing the wrong product increases.

- Changing requirements. When requirements change, the whole base for solution development changes. This causes unstable scope, changing concepts for implementation and even chaos. Changing requirements may be related to a changing business which is something quite natural for some business areas or domains and therefore cannot be avoided, but the reason of unstable requirements may be different – it may result from lack or poor quality of business goals for a given project. If we do not have clear business goals, in fact we do not know what we are going to achieve at the end – so, we do not know what we are doing.

- Unclear objectives. This is a very common problem in case of many IT (and not only IT) projects. Projects initiated without establishing business goals and objectives to be achieved as a result of that project. If there is nothing to guide the project and define its business deliverable, you don't know what are you doing and what for. If you don't know what are you doing, how can you propose any reasonable solutions? In other words – any solution will be wrong as there is no way to meet a goal, if the goal is not known.

Effects of the issues above are quite obvious. If we have problems at early stages of the project – and all the previous factors are in general related with early stages – we will have more problems later. As you all know, the later the problem is detected, the more it costs to fix it. In case of problems related with requirements or – what is even worst – business goals, the cost of problem resolution is much higher than fixing a simple bug in the code. Why? Because we will have to deal with something that is the base of the project.

What is the real problem?

Coming back to the Chaos report and the source of the problems – let's think which of the factors are really reasons of project failure? Are they problems or just symptoms of something else?

To answer this question, let's think what can we see in real life?

- There is no analysis and preparation for a project. We don't research the market, the user's needs, our own business processes. We just used to think: "Let's make a software" and then we are initiating the project. Projects are initiated without deep analysis, and determination of main goals, risks, benefits... The first step should be so called enterprise analysis [BABOK] where we are looking for business problems to be solved. This research includes business processes analysis, establishing business goals and needs, which in fact require knowing the organization strategy, weak and strong points, chances etc. In other words – at first we need to know AS IS and then, on this basis we are able to define TO BE, which forms a background for our project.
- Projects tend to deliver SOFTWARE, not SOLUTIONS. What does our customer want? A software? Really? The real aim of any project is not a software – but a solution resolving a given business problem and allowing to achieve specific goals. This solution may include software components, but we cannot just focus on software, as there are many other things that can a part of the final solution – like new business (products, services), changes of business processes, or procedures and many other, non-software related aspects.
- Main success criteria are usually time and cost. Too often we can see thinking like: if you deliver on time and within a budget, your project succeeds. It is not enough. There is something very important missing – quality.

To be able to say that we succeed, we need to achieve a specific level of quality. Even if you keep your project within the time and budget limitations, but you not provide the expected quality, you cannot say that the project was successfully completed. You may deliver on time and within a budget but the product you released is not the one that the customer needed and wanted.

Ok, but... What is this mysterious quality? There are many definitions, for the purpose of this discussion let's follow one, established by ISO 9000. The ISO definition says that "quality is degree to which a set of inherent characteristics fulfills requirements". This explanation is simple and expresses the main important aspect of quality – quality is determined by meeting specific requirements. You may say that it is a very subjective measure. Yes, indeed, it is. That's why we need to specify requirements in very clear and measurable (even numeric) way. Otherwise we will never be sure if the quality we deliver is the quality expected by the stakeholders.

Sounds quite simple, isn't it? The problem is how to assure quality if:

- Stated requirements are not complete?
- Stated requirements have no business value? In the end we will deliver product that do not bring any real value to the customer. We are just producing software for the sake of having software.
- We don't know how requirements meet business goals? We cannot measure how and even if our solution resolves given business problem.

So let's say it clear - the real problem we have in projects are requirements... Business requirements. We fail to define them in a way allowing to meet the stakeholders expectations. We fail to define them in a way that allows to ensure that the product deliver a value.

Writing Business Requirements

It is easy to say, let's write good requirements otherwise we will have problems in the later stages of the project. But how to do it, how to write good requirements? You can start from old, well known truths:

- Learn from mistakes. We know what mistakes we did. Avoid them.
- Follow best practices! There are known, checked, tested rules and principles to be followed. Use them.

Tom Gilb defined a set of ten rules to be followed when working with requirements. These rules are called Ten Key Principles for Successful Requirements. Let's discuss them now.

1. Understand the top level critical objectives

Let's think about our common experiences and ask ourselves how do we start projects? It there any serious research, analysis etc.? Not always, isn't it? In fact, very rarely. Many projects are initiated by just writing a few statements which we believe are business requirements. But these statements have no real background and sense if we do not define high-level requirements - the ones that come from the key stakeholders and create a base for the project. The ones that funded the project and are called as Top Level Objectives. These requirements should express what we want to achieve as a result of the project; they should express business goals. In most projects there are no high-level requirements at all – we start from requirements describing the solution itself. This way we missed an important aspect of the project – what are we going to achieve? Even if we have some high-level requirements, there is often another problem – they are often vaguely stated, and ignored by the project team. We – unfortunately – tend to start from describing, sometimes very detailed – elements of target solution. The first step should be to define and understand what is the business purpose and deliverables of the project.

How to do it and determine goals of the project in business terms? Think about the business and about the problem you want to resolve. For example, if you already have a system which was found to be hard to understand and use by users, the example of Initial Top Level Objective may be:

Make the system much easier to understand and use, than has been the case with the previous system

If you are working in banking and deal with the problem that completing a transaction takes too much time, the example of Initial Top Level Objective may be:

The solution will allow to perform core banking transaction in shorter time

Of course, such statements are not really measurable (what does it mean: "easier" or "in shorter time"?) therefore they should be refined by adding some detailed words making them more numerical and measurable (How much easier? Comparing to what? What is "shorter time"?).

2. Look towards value delivery

One of the most typical problems in IT is that we focus solely on producing software. Real business is not about any software. It is about systems, including information systems. Therefore you should change your way of thinking and think about system as a whole instead of focusing on software. Remember and recognize the fact that the main – and real – goal of a project is delivering realized value (benefits) to the stakeholders.

You should also understand and accept the fact that realized value is not the defined functionality! As defined by Tom Gilb, value is the benefit we think we get from something. So at first we need to determine the value that we are looking for. Again, sounds easy but is rather difficult to apply in real-life projects. Why? Because conventional requirements engineering is not closely enough coupled with "value" and therefore we have serious problems when trying to define "value". We used to focus on functions, attributes, screens and layouts of the final solution instead of thinking about the value we need to get. We tend to forget that we do not make projects for fun and for the sake of making software, but to get clearly defined benefit for the customers and sponsors. Moreover, to be able to ensure we deliver this benefit, it should be expressed in measurable terms as only then we are able to verify if the project really brought us with the expected value.

What if we miss this aspect of the process and do not define value? What if the requirements do not express value? Well, then we will have a problem. At first, we fail to deliver the value expected, even if „requirements" are satisfied. How is it possible? Well, we may just deliver product, which is compliant with requirements, but not useful for the stakeholders.

Another consequence is that if the requirements do not express value we may miss other things necessary to actually deliver complete value to stakeholders on time. Why? Because there is a risk we will not know that there are other things important or necessary to get the full value. For example – if you focus on software solutions, instead of thinking about the value and benefits, you may miss the fact that to be able to get the full benefit from using the software, a change of business process is also necessary. An example can be introducing a workflow system to support processing of documentation without analyzing and optimizing the whole process of documentation flow.

3. Define a „requirement" as a „stakeholder-valued end state"

We are talking about requirements but let's stop for a second and answer the question – what a requirement is? What it is for you, and what it is for your customer?

Before starting any project work you should ensure there is common, agreed and accepted definition of a requirement. You just need a glossary.

Tom Gilb proposed a definition describing a high-level requirement as a stakeholder-valued end state. It is important to notice that the focus is put (again) on value.

In addition, to ensure effective and transparent communication, define other terms you will use in the project:

- Requirement specification
- Solution
- Product
- Stakeholder
- Value
- Benefit
- Business goal

4. Think stakeholders: not just users and customers!

One of typical mistakes made when planning project works is missing important stakeholders. What's the problem, you might say. It is enough to know about the customer, business and users as they requested the product and pay for it. The problem is, it is not enough. Users and customers usually provides directly known and "obvious" requirements. But there are other important aspects and information affecting the project or product as well – competitor's data, market needs, limitations, and technology. This kind of information rarely comes as written requirements; it is usually discovered as a part of requirements elicitation and analysis. However to be able to collect this data we need to have someone to ask – we need stakeholders supporting requirements engineering works.

Therefore it is very important to remember that stakeholders are not only users or customers. There are many other stakeholders involved in every IT project. We should not focus requirements only on user or customer needs as in this case we may miss important needs, limitations or information coming from other sources. Deeper analysis requires broader area of stakeholders, their needs and values.

One of the basis definitions of stakeholders says that a stakeholder is anyone or anything that has an interest in the system. So, stakeholders are not just the end-users and customers, the following should be also considered: IT development, IT maintenance, senior management, government, regulation bodies, etc.

5. Quantify requirements as a basis for software engineering

If you can not measure it, you can not improve it – Lord Kelvin

What is our work about? It is about engineering, right? Software engineering. And what are we doing? Engineering? Not really... Real engineering is not about words as we used to do when working with requirements and specifications, but it is about numbers and measures. The problem we face in real IT projects is the lack of numeric quality requirements. Why? Because we don't know how to do it and how to practice real engineering in the context of software. We use words because no one teaches how to define requirements in numeric form.

Don't produce requirements specifications consisting merely of words as they will not be measurable and testable. They will not allow you to check if everything what was to be done is really done. There is a solution – you can just define a scale of measure to be used when describing requirements. You can follow Tom Gilb's approach or develop your own approach.

6. Don't mix ends and means

Albert Einstein said "Perfection of means and confusion of ends seem to characterize our age".

So true, isn't it? We mix end and means over and over again. We don't know what we want, but we are saying how to do things. Starting from the end is a common problem in IT. Why establishing the final result seems to be so difficult? Because solutions are more concrete. They are visible, we can see them, feel them, understand them. Qualities we want are more abstract. They require more analysis and thinking. We can define a solution faster and easier than establish a business goal, especially as working on business goals requires more business knowledge.

REQUIREMENT EXAMPLE

Usability.Intuitiveness:

Type: Marketing Product Quality Requirement.

Ambition: Any potential user, any age, can immediately discover and correctly use all functions of the product, without training, help from friends, or external documentation

Scale: % chance that defined [User] can successfully complete defined [Tasks] Immediately, with no External help.

Meter [Consumer Reports] tests all tasks for all defined user types, and gives public report.

Goal [Market = USA, User = Seniors, Product = New Version, Task = Photo Tasks Set, When = 2012] 80% ±10% <- Draft Marketing Plan

Fig. 1. *Example of a measurable description of a requirement, using Planguage [2][3]*

The problem is that to get what you want; you must first state what you want. Don't mix ends and means. Don't start with means as you will not get what you want. Don't specify a solution, design and/or architecture, instead of what you really want – real requirement.

Why not? It is easier, you may say. Yes, it seems to be easier but remember: "Be careful what you ask for, you might just get it". And you may be very surprised with this what you got….

If you specify a solution, not "what you really want" [2]:

- You might not get what you really want. If you haven't specified what you want, how can you expect you will get it?
- The solution you have specified might cost too much or have bad side effects, even if you do get what you want. Let's focus on the goals – leave the means for further analysis. That is why we have requirements analysis and solution designing activities to propose a solution design meeting you requirements in best possible way, with minimal risks. But there can't be any real requirements analysis if you already defined means…
- There may be much better solutions you don't know about yet. As above – if you state how to do what you want, you do not give a change to analyze your needs and propose a design of a solution meeting you requirements in best possible way.

Requirements should be written independent of the system that would be built to satisfy the need.

> *Instead: "The system must print a transaction receipt for the customer."*
>
> *Use: "The customer must be provided with a transaction confirmation after every transaction, within 2 minutes after completing a transaction."*

Great, but how to find out what you really want? There are many techniques and solution, but you can start with using simple technique called "5 x Why".

Search for the real need by asking "Why"? Let's consider an example. Imagine, your customer is asking you to provide him with a report. To get the answer what the customer really needs, you could initiate such conversation:

> You: Why do you want <the report>?
>
> Customer: Because I really want <specific data> and assume I will get it through this report.
>
> You: Then why do you want <specific data>?
>
> Customer: Because I really want <to calculate X> and assume that is the best way to get <the report>.

The simulation above is not as unrealistic as it may look at first glance. It is a quite often situation, that applying "5 x Why" technique allows to state that some requirements are not needed or are not requirements at all (instead, they could be for example, a part of the solution design).

7. Focus on the required system quality, not just its functionality

There are really not many projects aiming to deliver totally new products, most projects aim to improve operating of already existing solutions. In other words, quality improvements tend to be the major drivers for new projects. So keep in mind that what the system must do (functions) is important but don't forget about the important question on how well the system should perform (qualities). Focus on the quality requirements, rather than the functions as functions can be delivered quite easy, but to achieve required or expected level of operating, it is necessary to plan and develop certain quality characteristics.

These days the way of getting competitive advantage is delivering more useful, more reliable, more efficient solutions. Customers search for better products, and it is important to emphasize that the word "better" means something else for the customer, than for you. Qualities of a system are important factor that makes the customer happy or disappointed when working with the solution. It has been proven that quality requirements determine if the customer will like your product. You may have 2 products with exactly the same functionality but of different usability and the product of better usability will be perceived as a better product, than the second one. In case of websites, the user decides within the first 50 milliseconds whether or not he/she likes a website. Is it about functions? Not at all. You are not able to judge functions of any website within 50 milliseconds, so not the functionality decides about the user's perception of the website. It is something else: pleasant design, aesthetics, ergonomic aspects. These are all non-functional qualities. Let us consider another example. Imagine a navigation pane located in an aircraft cockpit. All buttons, indicators and major options must be very clearly marked and immediately accessible for a pilot. It is also about non-functional characteristics.

8. Ensure there is 'rich specification'

Another common problem is that far too much emphasis is often placed on the requirement itself. We used to focus on building and reading the requirement statement, missing other information allowing to understand the context and real meaning of the requirement. Usually there is too little concurrent information about the whole background describing for example, who wants this requirement and why, what benefits do we expect to get from the requirement.

When eliciting requirements it is necessary to collect higher level business data or other background information that provides the context of the solution. Sample background information suggested by Tom Gilb can be:

- Owner – who owns the requirement? It is especially important in case of conflicts or a need to explore details of the requirement.
- Version – the actual version of the requirement. This information should be handled as a part of version control processes.
- Stakeholders – who has any (positive or negative) interest in implementing this requirement? This knowledge is especially useful in requirements elicitation (as it indicates who should be

asked about the solution) and then, in requirements analysis, as it supports conflict management.

- Gist (brief description) – short summary expressing the most important aspects of the requirement.
- Ambition – what are we going to achieve? In other words, it is a statement of business goals. These goals should be expressed in measurable terms in order to allow further verification.
- Impacts – does the requirement have impact on other requirements? What can affect the requirement? This information allows to determine relationships and dependencies between requirements which is a base for further requirements analysis.

You may say that such information is nothing but unnecessary bureaucracy as it does not express the real content of the requirement. Indeed, it does not describe the requirement itself but it provides other information, necessary to understand what is the meaning and role of the requirement for the solution considered as a whole.

Background information provides the following benefits:

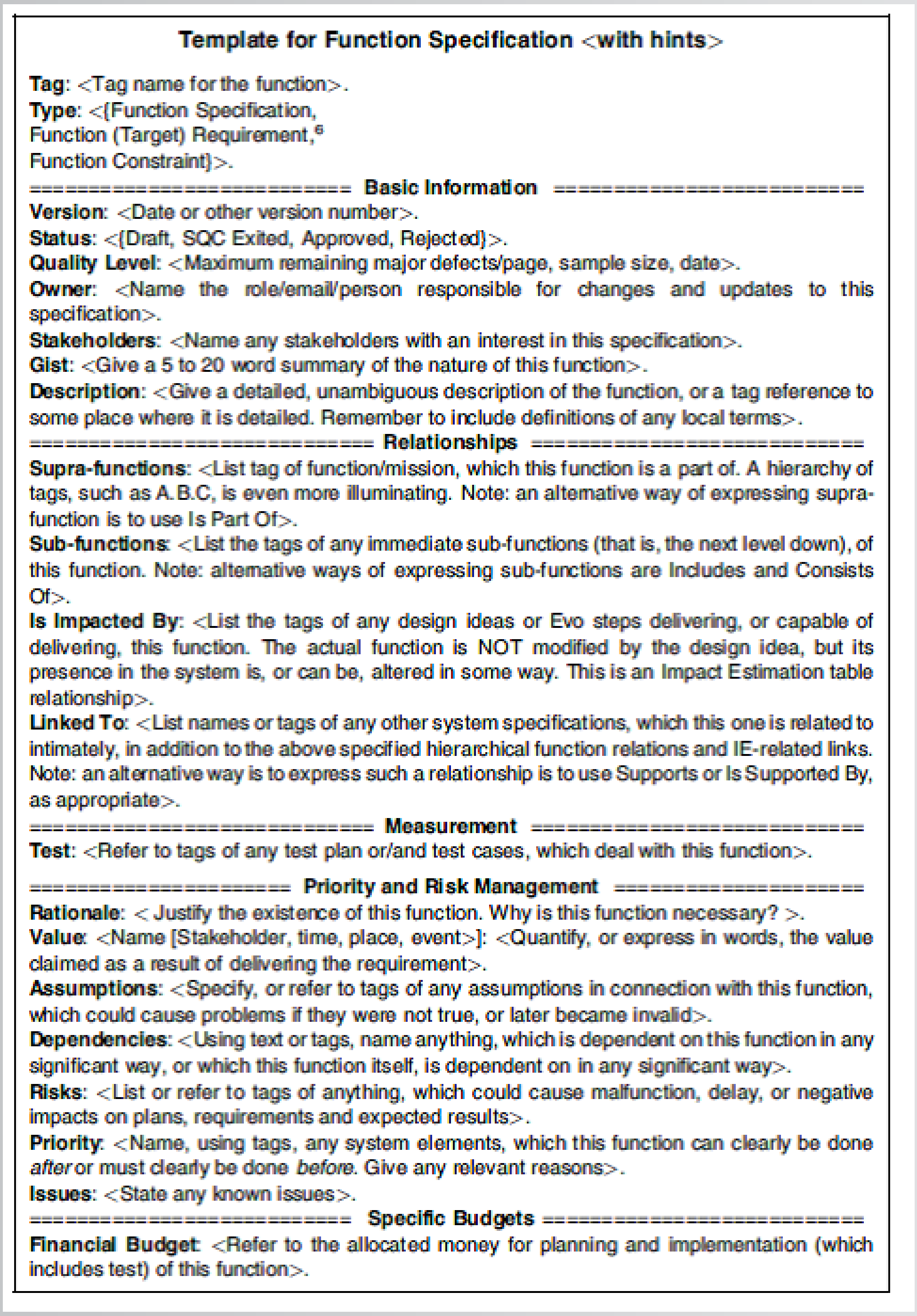
- It helps to judge value of the requirement
- It helps to prioritize the requirement and determine how important it is for the solution
- It helps to identify and understand risks related with the requirement
- It helps to update the requirement – additional information like impacts or relations can help to foresee potential impacts of a change.
- It helps to define and maintain the relationships between different but related levels of the requirements – information about relationships is usually expressed by traceability.
- It improves the clarity of the requirement – all the additional information serves to provide more detail about the context and relationships between requirements.

All the background information can be provided as part of the requirements specification. A sample template of such specification can be as follows (Fig. 2).

9. Carry out Specification Quality Control (SQC)

There is nothing strange or new in stating that requirements can have bugs. Requirements are written by people and humans can make mistakes. It is the same situation as in case of defects in the source code, resulting in product failures. Defects in the code are found by testing, either static or dynamic. How to find defects in requirements? By testing, too. We should also remember that the later defect is found, the more it cost. So it is quite obvious that we should start testing as soon as possible. Early testing means – testing requirements.

How to do this? We can check the quality of requirements against relevant standards; we can use quality control checklists based on quality criteria for requirements.



Template for Function Specification <with hints>

Tag: <Tag name for the function>.
Type: <{Function Specification,
Function (Target) Requirement,[6]
Function Constraint}>.
========================== **Basic Information** ==========================
Version: <Date or other version number>.
Status: <{Draft, SQC Exited, Approved, Rejected}>.
Quality Level: <Maximum remaining major defects/page, sample size, date>.
Owner: <Name the role/email/person responsible for changes and updates to this specification>.
Stakeholders: <Name any stakeholders with an interest in this specification>.
Gist: <Give a 5 to 20 word summary of the nature of this function>.
Description: <Give a detailed, unambiguous description of the function, or a tag reference to some place where it is detailed. Remember to include definitions of any local terms>.
============================ **Relationships** ============================
Supra-functions: <List tag of function/mission, which this function is a part of. A hierarchy of tags, such as A.B.C, is even more illuminating. Note: an alternative way of expressing supra-function is to use Is Part Of>.
Sub-functions: <List the tags of any immediate sub-functions (that is, the next level down), of this function. Note: alternative ways of expressing sub-functions are Includes and Consists Of>.
Is Impacted By: <List the tags of any design ideas or Evo steps delivering, or capable of delivering, this function. The actual function is NOT modified by the design idea, but its presence in the system is, or can be, altered in some way. This is an Impact Estimation table relationship>.
Linked To: <List names or tags of any other system specifications, which this one is related to intimately, in addition to the above specified hierarchical function relations and IE-related links. Note: an alternative way is to express such a relationship is to use Supports or Is Supported By, as appropriate>.
============================ **Measurement** ============================
Test: <Refer to tags of any test plan or/and test cases, which deal with this function>.

===================== **Priority and Risk Management** =====================
Rationale: < Justify the existence of this function. Why is this function necessary? >.
Value: <Name [Stakeholder, time, place, event>]: <Quantify, or express in words, the value claimed as a result of delivering the requirement>.
Assumptions: <Specify, or refer to tags of any assumptions in connection with this function, which could cause problems if they were not true, or later became invalid>.
Dependencies: <Using text or tags, name anything, which is dependent on this function in any significant way, or which this function itself, is dependent on in any significant way>.
Risks: <List or refer to tags of anything, which could cause malfunction, delay, or negative impacts on plans, requirements and expected results>.
Priority: <Name, using tags, any system elements, which this function can clearly be done *after* or must clearly be done *before*. Give any relevant reasons>.
Issues: <State any known issues>.
========================== **Specific Budgets** ==========================
Financial Budget: <Refer to the allocated money for planning and implementation (which includes test) of this function>.

Fig. 2. *Sample requirements specification template [3]*

A good practice is to apply the rule that all requirements and specifications should pass quality control checks before they are released for use by the next processes. This way we can minimize the risk of having serious problems later, when it appears that the requirements being basis for the solution design are ow quality. Testing requirements earlier gives a chance to find defects and correct them before starting any implementation works. So – we can avoid, or at least reduce the amount of reworking, regression and introducing additional risks resulting from late changes.

Some statistics indicate that initial quality control of requirements specification typically identifies 80 to 200+ words per 300 words of requirement text as ambiguous or unclear. This research involved checking against only three quality criteria, which are:

- Unambiguous to readers
- Testable
- No optional designs present

It is important to understand, that with no quality control performed, this number of defects would be passed to later phases of product developed.

To make quality control of requirements more complete, we may use checklist covering the following quality criteria:

- Correct – does it accurately describe the expected feature?
- Feasible – is it possible to implement within the estimated budget, time and limitations?
- Necessary – does it document what the stakeholders really need?
- Prioritized – does it have a priority defined and do we know how essential the requirement is?
- Unambiguous – can it be interpreted only in one way?
- Verifiable – is it possible to verify if it is implemented correctly?
- Singular – does one requirement statement describe only one requirement?
- Design independent – does it describe a need, not solution details?

10. Recognize that requirements change

The last principle says we should be aware that requirements may change and accept this fact. Requirements can evolve due to feedback from stakeholders, or because of changes resulted from the business. An example can be a need for change of a requirement caused by update of law or other regulation.

When thinking about changes consider factors from outside the system: politics, law, regulations, international differences, economics, and technology and/or business change.

Changes can always happen. Business is changing, new concepts may appear, or the current concepts are considered not good anymore. In real projects it is often not possible to avoid changes, as following the plan may lead to project failure in terms of not meeting its business goals.

We need to accept the fact that requirements may be a subject of change and there is nothing to stop it. All we can do with this is to make it easier to manage changes and reduce risks related to any modifications. How can we do it? One of the most important means to support changeability is implementing traceability between requirements and other project artifacts. This will help to analyze the impact of a change and to minimize the risk of introducing changes so that decisions about implementing changes can be made on reasonable basic and real estimates.

So, how to start...

We know the rules of writing good requirements. Some of them are not so easy to apply as it would require changing the whole mindset of the management, business and IT stakeholders; some would require re-organization of core business processes related to product development and even maintaining business strategy. However, we can start from implementing the basic rules.

Stakeholders

Let's start from the stakeholders. Before you can think about the requirements, you should ask yourself:

- Who will have any interest in the project itself or/and its deliverables?
- Who can be affected by realization of the project?
- Who can limit the capabilities?
- Who will be involved in project works?
- Who will be the management team?
- Who will be the business stakeholders?
- Are there any external bodies which can impact the solution?
- Are there any regulations or laws related to the business area covered by the project?

To be sure that requirements elicitation is complete you should know who is involved in establishing the business goals, scope, limitations and assumptions. Only then it is possible to minimize the risk of missing important information.

Business objectives

When all stakeholders have been identified, you can start with establishing what is to be done. It is recommended to start with determining business objectives what allows to provide a clear vision of what is to be accomplished.

Collect and understand business objectives. Understand the context of the organization, its dependencies and external and internal relationships with other entities. When the general high level objectives are already known, decompose them into smaller, S.M.A.R.T. goals. S.M.A.R.T. technique allows to define goals which are:

- Specific – a specific goal says precisely what should be done.
- Measurable – a measurable goal is expressed in numerical terms, so that you will be able to state if it was achieved or not.

- Attainable – an attainable goal is realistic and attainable in a given situation. So don't establish goals which are not possible to achieve!

- Relevant – establish goals that matter. Goals that are relevant to your management, your team, your organization will receive that needed support.

- Timely (time-bound) – round goals within a time frame, give them a target date. A commitment to a deadline helps a team focus their efforts on completion of the goal on or before the due date.

The following worksheet helps to establish and verify S.M.A.R.T. goals (Tab. 2).

Complete business goal statement would look like:

The sales team should increase sales of insurance products by 30% by the end of 2015, in the central sales region.

Derive business requirements

When you know the business goals, you can start with establishing business requirements. Business requirements can be understood as a further decomposition of business goals. Each business goal will be

S.M.A.R.T.

Goal					
Intention	**Specific**	**Measurable**	**Attainable**	**Relevant**	**Timely**
What do you want to achieve?	Who? What? Why? Where? When?	How much? How often? How many?	Achievable?	Is it important to achieve it?	By when?
Increase sales	The sales team. Insurance products. In the central sales region.	By 30%.	Yes.	Yes.	By the end of 2015.

Tab. 2. *Establishing S.M.A.R.T. goals.*

EXAMPLE

Business goal:

Increase incomes from selling insurance products to 50.000 Euro by the end of 2015.

Requirements allowing to meet the specific goal may be related to:

Improving the efficiency of selling process (for example, by having up to date information about new products and services)

Opening new selling channels (for example, internet)

Creating a method for monitoring current incomes against the plan.

Instructing the insurance agents in new processes and products.

implemented by several business requirements. Remember that the requirements are not necessary related to any software! So don't think in software terms – think in business terms.

When establishing business requirements, ensure the owner of each requirement is allocated and informed about his/her role for the later activities related to solution design. When there is a specific person or group responsible for a requirement, it is much easier to work on accomplishing it.

It is important to "think business" and do not focus on software only. In fact, deriving business requirements should be concerned with answering the question: "what do we need to have to meet business goals".

The requirement statement

Write the statement expressing the requirement. You can use the following structure of the requirement statement:

- The user - who would like this requirement?
- The result - what is the result they are looking for?
- The object - what is the object the requirement addresses?
- The qualifier - what is the qualifier that is measurable?

For example:

The insurance agent must have information about any new products one day prior to the product launch.

Let's look at the structure of the business requirement.

The insurance agent <-- who

must have information about <-- what result

any new products <-- what object

one day prior to product launch <-- qualifier

Remember not to determine the solution – just state what needs to be achieved. State WHAT, not HOW. There will be a proper time for defining the solution. Now you just want to state what you need.

Traceability

To be sure that all business objectives are met, link each requirement with appropriate business goal. Use traceability to show you are performing all necessary steps in the process – starting from the identification of business goals, through business requirements, to design artifacts. When decomposing business requirements into solution requirements, link each solution requirement with appropriate business requirement. Remember that business goals are a kind of acceptance criteria for a product – they will serve as a basis for assessment, so you must be sure they are considered on different levels of solution development. Traceability allows ensuring that products of specific level implement artifacts of the previous level.

Example of traceability between a business requirement and solution requirements (use cases) is presented on the figure (Fig. 3).

TRACEABILITY

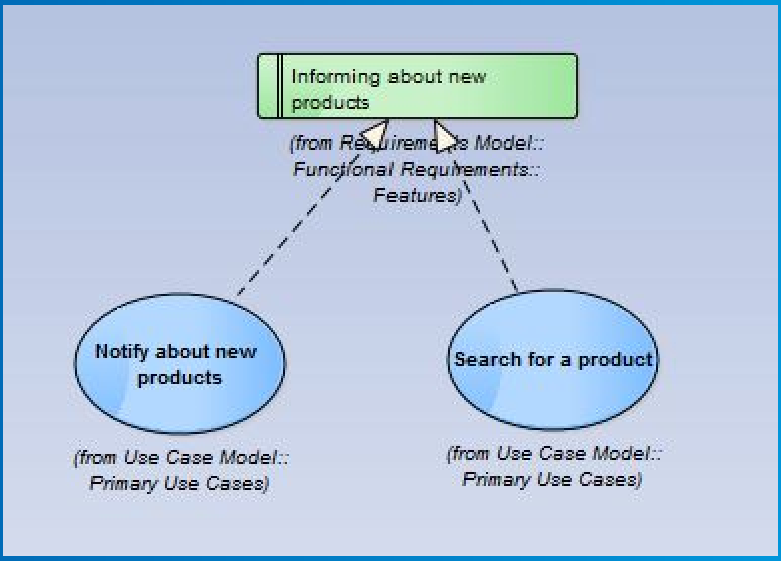

Informing about new products
(from Requirements Model:: Functional Requirements:: Features)
Notify about new products
(from Use Case Model:: Primary Use Cases)
Search for a product
(from Use Case Model:: Primary Use Cases)

Fig. 3. *Visualization of traceability*

Quality Control

Remember to continuously check quality of the requirements. Use metrics, check lists and standards to ensure the requirements are of good quality. Plan reviews aiming to verify completeness, correctness and consistency of products of your works.

Summary

As we know the impact of business requirements on the success of any IT project cannot be neglected. Poor requirements cause problems. Sometimes these problems lead to failure of the project. You know what can go wrong, so don't do things that are risky. Do what should be done in order to minimize the risk.

Establish business goals so that you know what should be done. Link business requirements with goals. State requirements in measurable way and ensure they express stakeholder's value. Think who wants what, not how. And don't forget about quality control as to minimize the risk of project failure knowing the current state of the product is essential.

REFERENCES

[1] http://www.cafe-encounter.net/p1183/it-success-and-failure-the-chaos-report-factors

[2] Gilb, Tom. What's Wrong With Requirements.

[3] Gilb, Tom. Planguage Concept Glossary.

[4] Gilb, Tom. Competitive Engineering: A Handbook for Systems Engineering, Require-

AUTHOR

Karolina Zmitrowicz

Karolina currently works as independent IT consultant and trainer in Requirements Engineering and Quality Assurance fields.

She has a strong experience in the fields of requirements engineering, project management, quality assurance and testing. During her career she worked as Project Manager, QA Manager, Change Manager, Technical Writer, Business and System Analyst, ISO 9001 Consultant, IT Consultant and Trainer. She has international experience in banking and insurance sector: she worked for leading financial organizations in South Africa, Netherlands, Austria, Slovakia, Italy and Poland.

She is also an author of several publications in Software Engineering area.

Maciej Chmielarz

Hash verification in regression testing

INTRODUCTION

In regression testing it is quite common to compare results obtained with the new version of software to reference results that were proven to be correct in previous processing. This task can be challenging when straightforward conformity is broken by mismatches on non-essential data, like automatically incremented primary keys, current dates etc.

Unfortunately in most of the cases record ids can't be ignored, because we need to keep track of dependencies between tables. Resetting them to some initial values doesn't always help either. Looking for universal solution to this issue I thought of replacing keys with values (almost) unequivocally bound to essential content of each record - a hash value counted upon it.

RESULT 1

```
account
 a_id | name   | reg_date    | credits
------+--------+------------+---------
 101  | Hewey | 2014-02-12 | 400
 102  | Dewey | 2014-02-12 | 320
 103  | Louie | 2014-02-12 | 100


station
 s_id | loc_name
------+--------------
 201  | Spoke Square
 202  | Chain Street
 203  | Gear Avenue
 204  | Fork Street
 205  | Pedal Square


rental
r_id | a_id | pickup_st | return_st | pickup_ts           | return_ts           | credits
-----+------+-----------+-----------+---------------------+---------------------+--------
301  | 102  | 204       | 201       | 2013-05-15 08:45:12 | 2013-05-15 08:57:35 | 10
302  | 101  | 202       | 205       | 2013-05-15 10:11:56 | 2013-05-15 10:37:08 | 20
303  | 101  | 205       | 201       | 2013-05-15 12:32:24 | 2013-05-15 12:48:10 | 20
304  | 103  | 203       | 202       | 2013-05-15 15:03:21 | 2013-05-15 16:12:38 | 50
305  | 102  | 201       | 204       | 2013-05-15 17:10:43 | 2013-05-15 17:26:23 | 20
306  | 102  | 204       | 201       | 2013-05-16 08:47:21 | 2013-05-16 08:58:05 | 10
307  | 103  | 202       | 205       | 2013-05-16 13:15:00 | 2013-05-16 14:32:00 | 60
```

The method

In the proposed method first we determine a specific hash value for every record of every table being analyzed. Then we store those hashes in an auxiliary table. Full content of the auxiliary table can be later used to check conformity, but in the meantime hashes can also substitute keys while counting hashes for records in dependent tables.

Let's see an example. Assume that in our sandbox database we process data from automated bike rental system that requires its customers to buy prepaid credits and holds information about where and when bikes were picked up and returned.

Input data processed on 2014-02-12 gave following result (Result 1).

The same input data processed on 2014-03-02 gave following result (Result 2).

Although both results are formally the same, simple text comparison shows plenty of differences (you can check that copying them to any online difference checking tool). That way it is not easy to determine the test result. But if we get rid of all ids and current dates by hashing essential content, in the simplest case meaning all except ids and values of current date fields, what we get is what follows.

For the first processing on 2014-02-12:

RESULT 2

```
account
 a_id | name  | reg_date   | credits
------+-------+------------+---------
 101  | Dewey | 2014-03-02 | 320
 102  | Hewey | 2014-03-02 | 400
 103  | Louie | 2014-03-02 | 100


station
 s_id | loc_name
------+--------------
 206  | Chain Street
 207  | Fork Street
 208  | Gear Avenue
 209  | Pedal Square
 210  | Spoke Square


rental
 r_id | a_id | pickup_st | return_st | pickup_ts           | return_ts           | credits
 -----+------+-----------+-----------+---------------------+---------------------+-----
 308  | 101  | 207       | 210       | 2013-05-15 08:45:12 | 2013-05-15 08:57:35 | 10
 309  | 102  | 206       | 209       | 2013-05-15 10:11:56 | 2013-05-15 10:37:08 | 20
 310  | 102  | 209       | 210       | 2013-05-15 12:32:24 | 2013-05-15 12:48:10 | 20
 311  | 103  | 208       | 206       | 2013-05-15 15:03:21 | 2013-05-15 16:12:38 | 50
 312  | 101  | 210       | 207       | 2013-05-15 17:10:43 | 2013-05-15 17:26:23 | 20
 313  | 101  | 207       | 210       | 2013-05-16 08:47:21 | 2013-05-16 08:58:05 | 10
 314  | 103  | 206       | 209       | 2013-05-16 13:15:00 | 2013-05-16 14:32:00 | 60
```

```
hash
 table_name | id  | hash
------------+-----+----------------------------------
 account    | 101 | acb01b341996fb7715d51b0974c96c3d
 account    | 102 | c0254c5d234e28a60b15c606c26d4b35
 account    | 103 | fefa7c15b785f0497d79386041429b37
 station    | 201 | 2dd15de273dd713a6019831385ef5689
 station    | 202 | 6b97b13923f2c153e4e8ded7d229d020
 station    | 203 | 8d3d50bdfb72540174e2bdbf23a72466
 station    | 204 | f32bceb33574fc9b50c4a5155fbace42
 station    | 205 | 66a6722e8c122fea311cb624cc595700
 rental     | 301 | 0e13d8c81ad0236872247310ae58b6ba
 rental     | 302 | 336169d78681b93c2719deafeb0f9b5e
 rental     | 303 | 088c385ae37c35d631771497e7ccd694
 rental     | 304 | 70173bd7107ae3cccdb83bb81ee7f488
 rental     | 305 | a49ed327ddb847b8bab528ffaa707245
 rental     | 306 | f03881be73c11d4b3a5edbe5bac11a75
 rental     | 307 | 034130395868e5053339368f7759043c
```

...and for the second processing on 2014-03-02:

```
hash
 table_name | id  | hash
------------+-----+----------------------------------
 account    | 101 | c0254c5d234e28a60b15c606c26d4b35
 account    | 102 | acb01b341996fb7715d51b0974c96c3d
 account    | 103 | fefa7c15b785f0497d79386041429b37
 station    | 206 | 6b97b13923f2c153e4e8ded7d229d020
 station    | 207 | f32bceb33574fc9b50c4a5155fbace42
 station    | 208 | 8d3d50bdfb72540174e2bdbf23a72466
 station    | 209 | 66a6722e8c122fea311cb624cc595700
 station    | 210 | 2dd15de273dd713a6019831385ef5689
 rental     | 308 | 0e13d8c81ad0236872247310ae58b6ba
 rental     | 309 | 336169d78681b93c2719deafeb0f9b5e
 rental     | 310 | 088c385ae37c35d631771497e7ccd694
 rental     | 311 | 70173bd7107ae3cccdb83bb81ee7f488
 rental     | 312 | a49ed327ddb847b8bab528ffaa707245
```

CODE 1

```
insert into hash
select ,account' as table_name,
  a_id as id,
  md5(
    coalesce(char(name),'null') ||
    case
      when reg_date = current_date then ,current_date'
      else reg_date
    end ||
    coalesce(char(credits),'null')
  ) as hash
from account;

insert into hash
select ,station' as table_name,
  s_id as id,
  md5(
    coalesce(char(loc_name),'null')
  ) as hash
from station;

insert into hash
select ,rental' as table_name,
  r_id as id,
  md5(
    -- a_id
    coalesce((select hash from hash
                where id = t.a_id
                and table_name = ,account'),'null') ||
    -- pickup_st
    coalesce((select hash from hash
                where id = t.pickup_st
                and table_name = ,station'),'null') ||
    -- return_st
    coalesce((select hash from hash
              where id = t.return_st
              and table_name = ,station'),'null') ||
    coalesce(char(pickup_ts),'null') ||
    coalesce(char(return_ts),'null') ||
    coalesce(char(credits),'null')
  ) as hash
from rental as t;
```

```
rental      | 313 | f03881be73c11d4b3a5edbe5bac11a75
rental      | 314 | 034130395868e5053339368f7759043c
```

When we select just table names and hashes and sort the result in alphabetical order, we get exact match. That makes us quite certain that new results match reference ones. But before we can make use of benefits brought by this solution, first we need to run some queries to obtain our hashes (Code 1).

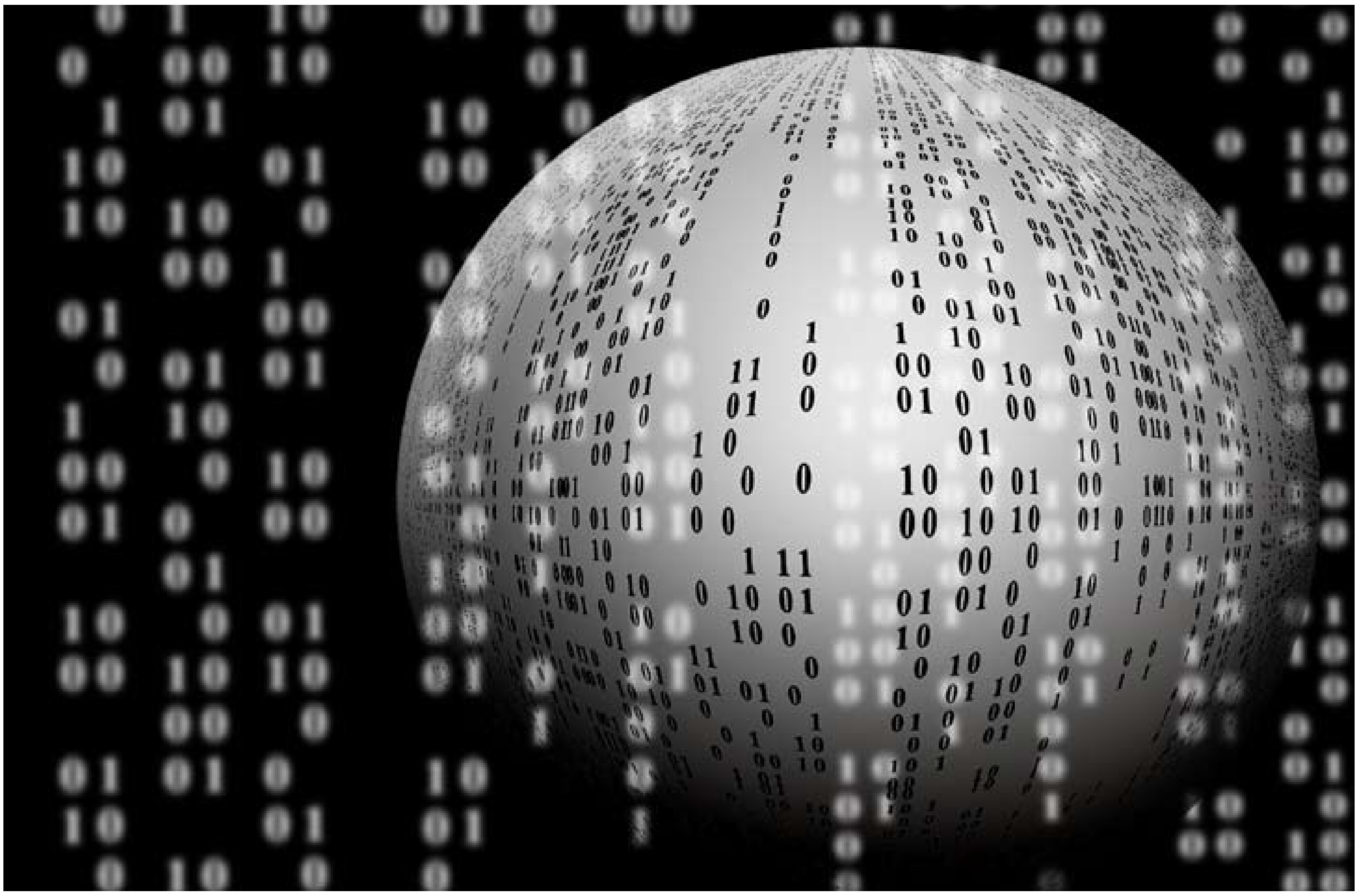
Let's take a closer look at some lines taken from the SQL code.

```
   coalesce(char(name),'null') ||
```

We use char function to convert value into string, regardless of original column type, because converting string into string makes no harm and it is easier to convert everything than to analyze if we need to convert or not. Two pipes at the end of the line mean concatenation with the value in next line. At the end the whole long concatenated string goes as an argument into MD5 hash function.

We use coalesce function to avoid null values. We need to check for nulls because concatenating anything with a null value results in a null value and all information is lost. Coalesce takes two arguments and checks if the first one is null - if it is not, returns the first argument, if it is, returns the second argument, which in this case is the 'null' string.

```
   -- a_id
   coalesce((select hash from hash
               where id = t.a_id
                  and table_name =
'account'),'null') ||
```



Column rental.a_id references account.a_id and to keep the result independent from specific key values, before we calculate hashes for records from rental table, we need to replace rental.a_id with appropriate hash extracted from the auxiliary hash table. We add coalesce to avoid consequences of situation when subselect doesn't find what we need and returns null.

```
    case
       when reg_date = current_date
then 'current_date'
      else reg_date
    end ||
```

Case function lets us check if the value that we expect to be the current processing date indeed is. If so, then we put 'current_date' into concatenated string. That way specific date will not affect the result when the concatenated string is hashed.

After building up the record consisting of table name, original record's identificator and respective hash value we insert it to the auxiliary hash table for future reference and final simplified conformity check.

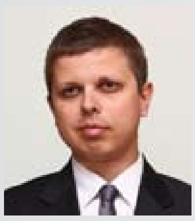
Summary

Described solution has its disadvantages. Composing SQL queries does take some significant amount of time - admittedly they can be generated based upon the database schema, nevertheless always need some manual tuning. That is why this method works best for systems with mature and invariable schema. Besides we need to be aware that we check solely the data that is being queried. But those drawbacks are no different than in case of other automated methods.

AUTHOR

Maciej Chmielarz

I have been testing software since 2008. I am a tester that meticulously and relentlessly traces anomalies in software development process. Currently I work as a Quality Assurance Engineer for IVONA Software, an Amazon Company that developes state of the art text-to-speech solution. Previously I worked for Asseco Poland on the biggest IT project in Eastern Europe and I started my career in Gdansk office of Acxiom Corporation. I am a lecturer and specialization supervisor at postgraduate studies on software testing at Gdansk School of Banking. After hours I organize rallies and other motorsport events.



Contact me: maciej.chmielarz@gmail.com

6th World Congress for Software Quality

1st - 3rd July 2014 Hilton Metropole, Edgware Road, London

"Shift left - inspiration and innovation for software quality"

The Sixth
WCSQ
World Congress for Software Quality
1st - 3rd July 2014
www.wcsq.org
London
"Shift left - inspiration and innovation for software quality"

Bringing together the world's leading experts in software quality in one of the world's most iconic cities
BOOK NOW
Earlybird discount ends 31st May
WATERLOO BRIDGE

Łukasz Gałuszka
Stefania Winkel

TestingCup

The idea

The idea of massive competing was unique as of that day. This is the story about the group of enthusiasts who converted the simple concept into the most important testing event in Poland.

The idea of organising Testing Cup started in Poland as early as 2011. Radek Smilgin became the originator and the main organizer of the idea. A few volunteers cooperated with him in this project. The beginnings were difficult, the aim was far away but every new day and every new team member took the championship to a new level. The result of their work were first Polish Championships in Software Testing – TestingCup. The event took place on 23th September 2013, at The National Stadium in the capital of Poland, Warsaw, the stadium where the Euro 2012 matches were played.

However, coming back to the beginnings, as the team was scattered across Poland, the organisation meetings ran online to the late night hours. There was a lot of work to do, starting from setting up the rules of the event, through searching for sponsors, to finally writing and testing an application intended for tests on the championships – Mr Buggy. Additionally, to rise the work pleasure over the project and get to know each other better the team members were meeting in Katowice or Wrocław. The friendly relationships developed better cooperation during the event.

The application created – Mr Buggy is a type of tool known by all testers – defects tracking tool. The role of the team was to create this application and to seed the incidents in it, which had to be found by participants. Only a few people hadseen and tested Mr Buggy before the TestingCup day. Most of them were nominated to the TestingCup Committee.

Realization

The preparations at the stadium started at 5 a.m. on the day of the competition. 120 computers for all participants had to be set, run and checked. The sounds of the starting operating systems accompanied us for half an hour. The event began according to the timeline, the reception worked ideally and with positive attitude. The registered participants and the spectators who came to see the work of the potential masters chose the comfortable places and went for coffee. The subject of the application was quite surprising for the participants. They knew they should be looking for bugs in the software but they did not know any details about the application usage. The competition task sounded: "Please, find and report defects, using the provided documentation, in the Mr Buggy application which is used to… incidents reporting". The participants reaction was obvious – smiles appeared on their faces.

After the official beginning, the volunteers helped to resolve all problems related to hardware. The spectators were able to see the participants as well as to take part in the lectures prepared by the eminent specialists in the field of testing and quality management in IT projects. The main speaker was David Evans who talked about agile testing.

The task for the participants was to find as many bugs as possible basing on the provided specification and report them in the text file. There was an additional challenge for testers – "what kind of information is important in the incident report, what is worth of reporting and what is a waste of time?". Some of them managed the task perfectly. They created reports which contained all the necessary information needed to the reproduction of the defect by the TestingCup Committee. The others, unfortunately lost valuable points for the quality of the report – not everyone can be a champion. :)

Punctually at 5 p.m., after 4 hours of heated debates, the TestingCup Committee chose winners. The results were announced solemnly and the awards were given to the first Polish champions in testing in individual and team categories.

Evaluation and plans

The event was rated very positively by Polish testers community. A lot of participants could face software testing process first time. The places for competition sold out at once – in 15 minutes! Organisers themselves were surprised by such a big interest.

The second edition of the cup is taking place at the beginning of June 2014. As a result of the experience gained in the first edition, some changes have been planned in some aspects. The event will last two days, we expect more participants – 200 testers and 100 spectators, bigger workspace was rented on the National Stadium and greater emphasis has been placed on the testers community integration. All that for the main purpose of the championship which is software testing promotion. More speakers will participate in this year's edition. The main speaker will be Paul Gerrard, a great authority in the field of testing, author of many books, adviser and tutor. He will talk about testing values. The second speaker will be James Lyndsay who will talk about exploratory testing.

For more details about the event, please visit the official website: www.testingcup.com.

AUTHORS

Stefania Winkel

Stefania Winkel is a graduate from Wroclaw University of Technology, Faculty of Electronics. She has been part of international software testing project since 2010. She has got The ISTQB Certified Tester Full CTAL.

Started with manual testing, designing test strategy, till test management – where there's a will there's a way. Her passion is a test promotion. She is a trainer preparing testers for ISTQB exam and a volunteer in Polish Championship in Software Testing "TestingCup".

Łukasz Gałuszka

Łukasz Gałuszka is a tester and test manager with a 10-year-experience within the branch. He works in the company which produces software for organizations delivering solutions for telecommunications network providers. He tested computer applications, mobile phones, elements of telecommunications networks as much as programmers' patience.

An accredited ISTQB Foundation Level trainer. A volunteer and a member of the TestingCup Committee since the first edition.

QUALE

Magazine

Publisher
VWT Polska Michał Kruszewski
Przy Lasku 8 lok. 52, 01-424 Warszawa
Number NIP 5272137158
Number REGON 142455963

Chief editor
Karolina Zmitrowicz
karolina.zmitrowicz@quale.pl

Deputy chief editor
Krzysztof Chytla
krzysztof.chytla@quale.pl

Editors
Bartłomiej Prędki
bartlomiej.predki@quale.pl
Michał Figarski
michal.figarski@quale.pl
Piotr Poznański
piotr.poznanski@quale.pl

WWW
www.qualemagazine.com
www.quale.pl

Facebook
http://www.facebook.com/qualemagazine

Advertisement
info@quale.pl

Cooperation
If you are interested in cooperating with us, please send us a message:
info@quale.pl

The following graphics have been used:

Cover

black_hole_space_outer_space
http://images.all-free-download.com/images/graphiclarge/black_hole_space_outer_space_216348.jpg