Food for Thought™

An e-newsletter published by
Software Quality Consulting, Inc.

January 2009 , Vol. 6 No. 1
[Text-only Version]

Welcome to ***Food for Thought***™, an e-newsletter from **Software Quality Consulting**. I've created free subscriptions for my valued business contacts. If you find this newsletter informative, I encourage you to continue reading. Feel free to pass this newsletter along to colleagues by clicking this **Forward Email** link. If you've received this newsletter from a colleague and would like to subscribe, please click this **Enter New Subscription** link. If you don't wish to receive this newsletter, click the **SafeUnSubscribe**™ link at the bottom of this newsletter, and you won't be bothered again.

Your continued feedback on this newsletter is most welcome. Please send your comments and suggestions to **info@swqual.com**.

In this issue

In **This Months' Topic**, I discuss techniques for risk-based testing...

Regular features to look for each month are:

- **Monthly Morsels**
  Hints, tips, techniques and reference info related to this month's topic
- **Calendar**
  Conferences, workshops, and meetings of interest to software engineers, QA engineers and anyone interested in software development

This month's topic

Risk-based Testing

Managing risk is a challenge for every project team. Some project teams ignore risk and hope it won't affect them - it always does. Other project teams take a more proactive approach by identifying and managing risks and coming up with possible mitigations - a much more realistic and effective approach.

My earlier discussions on risk (**June 2008** and **Sept 2008**) identified many different kinds of risk that project teams typically encounter. These risks may include both **internal risks** as well as **external risks**.

Internal Risks [1] may include:

- **Schedule Risks**
    - Is the schedule realistic?
    - What assumptions were made in developing the schedule?
    - Are all the resources identified in the schedule available from the start?

    Learn more about Estimating and Scheduling Best Practices...

- **Staffing Risks**
    - Are the best people available?
    - Do they have the right skills for this project?
    - Are enough people with right skills available?
    - Are people committed for the duration of the project?
    - Have staff members received necessary training?
    - Will turnover likely affect the project?

- **Process Risks**

- Are requirements well defined and unambiguous?
- Is there a documented development process?
- Does Management support following the development process?
- Is the development process followed?
- Are project software standards followed?
- Are peer reviews part of the development process?
- Has everyone been trained in peer reviews?
- Are CM tools, procedures, and training in place?
- Is there a process for changing requirements?

Learn more about Writing Requirements...

Learn more about Peer Reviews and Inspections...

- **Technology Risks**

  - Is technology new to your organization?
  - Are new algorithms required?
  - Does software interface with new or unproven hardware?
  - Does software interface with unproven 3rd party software?
  - Are there unreasonable performance requirements?

External risks include:

- **Risks to society**

  In 2004, the US Food and Drug Administration (FDA) recalled an infusion pump - a device used in hospitals to regulate the dosage of intravenous medication. The recall was initiated after several patients died as a result of receiving over doses of medication. An investigation revealed that defective software in the device allowed the dosage time information (hours and minutes) to be interchanged thus leading to the over dosage.

Learn more about reducing risks of medical device software...

- **Financial or economic risks**

  In 2003, defective software was a major contributor to the Northeast power blackout, the worst power system failure in North America. Over 50 million customers lost power as 100 power plants were shut down. Financial losses from this failure were estimated at **$6 billion**. [3]

  Learn more about Software Verification and Validation for Mission-critical Systems...

- **Political risks**

  Software has even been involved in politics. The public's confidence in electronic voting machines has been tainted due to poor design and ignored risks. A recent controversy in California highlights this issue:

  > "California Secretary of State Debra Bowen announced on Friday that the state hopes to recertify and continue using electronic voting machines produced by Diebold, Sequoia, and Hart, even though the machines have known security vulnerabilities and severe flaws. The state government decided that the machines can still be used as long as the vendors adhere to a lengthy list of requirements that aim to limit the potential for security breaches and machine failure.
  >
  > This announcement from the state follows extensive red team security audits that illuminated profound security failings in all of the electronic voting machines that were subjected to scrutiny. The security researchers, who analyzed the voting

> machines found ways to modify firmware, gain root access, trivially circumvent voting machine physical security mechanisms, install self-propagating trojan horses, and manipulate mock elections. On Diebold's voting machine, which uses the Windows operating system, researchers even found a remotely-accessible administrative account that wasn't protected by a password." [4]

Risk-based testing is yet another attempt to try to focus the testing activity in areas that provide the most value to customers and development organizations.

What kinds of risks are we talking about?

In the context of risk-based testing, we want to identify areas of **risk to your customers**. Some examples:

- **Customers** have problems installing your newly-released application
- **Customers** report problems with using new features - they don't seem to work as they had expected them to
- **Customers** report data integrity problems
- **Customers** have performance problems

Risk-based testing is all about identifying and managing risks that could negatively affect your customers when they use your software.

How can project teams identify these risks? There are a couple of tools and techniques available to help...

What do t-shirts have to do with risk?

Given that risk-based testing is focused on risks that could negatively affect your customers, there is likely much discussion and disagreement within the project team as to what exactly those risks

might be... here's where the t-shirts come in.

T-shirt sizing is a very effective estimating technique developed by Steve McConnell [2]. We can use this technique to help identify risks as well. Here's how...

> Key members of the Project Team - representing Marketing, Development, QA and Customer Support identify the most critical risks they see from the customer's perspective. These risks are put into a table and each group is asked to rank all of them using t-shirt sizes - small, medium and large as illustrated below. Once the risks are ranked, the group convenes and discusses them. At the end of the discussion, new risks may be identified and changes to the rankings may be made. The process iterates until the team agrees on the risks and their rankings.
>
> The most critical risks are identified and these are then the risks that testers focus on in their testing.

Risk	Marketing	Development	QA	Customer Support
Risk A	**Large**	**Small**	**Medium**	**Medium**
Risk B	**Small**	**Large**	**Large**	**Large**
Risk C	**Large**	**Small**	**Medium**	**Medium**
Risk D	**Medium**	**Medium**	**Small**	**Small**

Risk E	**Small**	**Small**	**Medium**	**Medium**

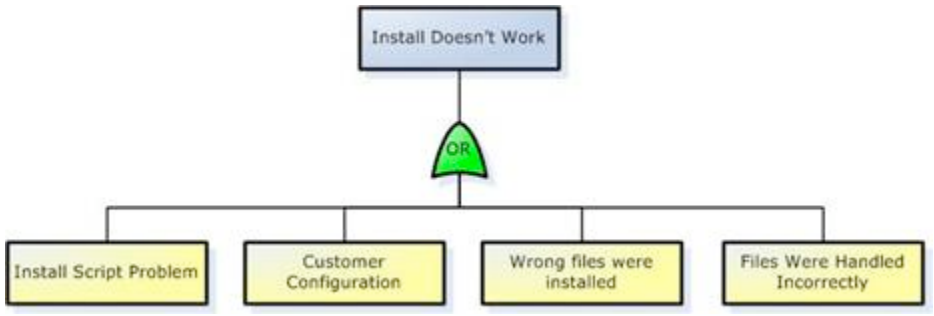
Fault Tree Analysis (FTA)

A fault tree is a tool that can help uncover potential customer risks in your applications before they are released to customers. To use FTA you will need a small team of people - ideally one developer, one tester, one customer service person and a facilitator (a.k.a. Project Manager). Here's what happens:

The FTA Team comes up with a list of say 5 or 10 of the **worst possible things that could happen when your customers try to use your new application.** These could be things like:

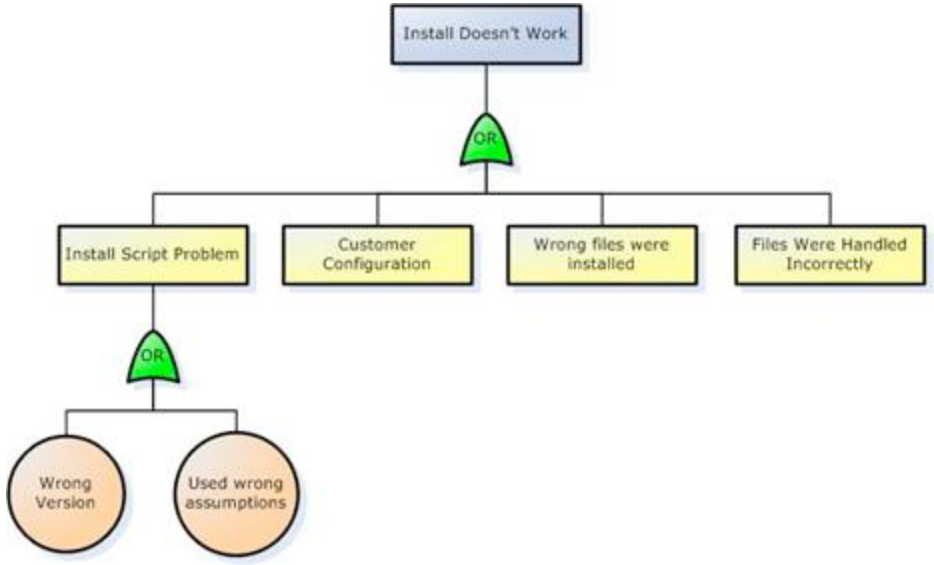
- The install doesn't work
- The migration process from the previous release doesn't work
- Customers are not happy with performance
- Once installed, the system crashes often...
- etc.

The team then creates a Fault Tree for each item on the list. Each item is placed at the top of the fault tree and then the team asks - How can this happen? Then they identify ways that the item at the top can happen. Here's a simple example:



In this example, the team said that there are four ways this could happen. Once they are satisfied that they haven't overlooked anything, they pick one of these and decompose it further, by asking

again “How can this happen” as shown below...



The team determined there are two ways that could cause an Install Script Problem - one being the wrong version of the script was included in the release and the other being that some of the assumptions used in creating the install script were wrong. You will note that Wrong Version and Used wrong assumptions appear in a circle. This means that they are basic events - in other words, they can’t be further decomposed. Often there may be a few intermediate levels between the top row and the basic events...

This process is repeated until all of the branches of the tree are decomposed down to their basic events. Once the tree is completed, the testers now have several things they should test to help reduce the risk that the Install Doesn’t Work...

Learn more about Risk Management using Fault Tree Analysis...

The Bottom Line

Risk-based testing can add value to the testing activity by

dramatically reducing the likelihood that customers will encounter the risks you identify. Clearly, the team identifying these risks needs to have enough domain knowledge to really understand what your customers do with your software.

Risk-based testing should be one of several different testing methods used on projects. These include:

- Requirements-based testing
- Scenario testing
- Act Like a Customer Testing™
- Risk-based testing
- Exploratory testing

The wider the variety of test methods used, the more effective your testing will be.

'Til next time...

Every month in this space, you'll find additional information related to this month's topic.

References

1. Pressman, R., *Software Engineering: A Practitioner's Approach*, McGraw-Hill, 1997, 4th ed.

2. McConnell, S., *Software Estimation – Demystifying the Black Art*, Microsoft Press, 2006

3. **2003 Northeast Power Blackout affects 50 million people**

4. Paul, R., "**California to recertify insecure voting machines**", *ars technica*, August 2007.

Additional Resources

1. **Fault Tree Analysis basics**

2. Lister, T. and DeMarco, T., *Waltzing With Bears: Managing Risk on Software Projects*, Dorset House, 2003.

Calendar

Every month you'll find news here about local and national events that are of interest to the software community…

- **Software Quality Calendar**

  There are many organizations that sponsor monthly meetings, workshops, and conferences of interest to software professionals. **Find out what's happening...**

- **Workshops Offered by Software Quality Consulting**

  Software Quality Consulting offers workshops in many topics related to software process improvement. **Get more info...**

About SQC

Software Quality Consulting provides consulting, training, and auditing services tailored to meet the specific needs of clients. We help clients fine-tune their software development processes and improve the quality of their software products. The overall goal is to help clients achieve Predictable Software Development™ – so that organizations can consistently deliver quality software with promised features in the promised timeframe.

To learn more about how we can help your organization, **visit our web site** or **send us an email**.

I hope this newsletter has been informative and helpful. Your comments and feedback are most welcome. **Send me your feedback...**

Thanks,

Steve Rakitin

info@swqual.com

Software Quality Consulting Inc.

Steven R. Rakitin
President

- Consulting
- Training
- Auditing

Phone: 508.529.4282
Fax: 508.529.7799

www.swqual.com
info@swqual.com