**INMM 2006 Oral Presentation – Paper Number 268**

# Tools for Trainers: Secrets of Safeguards Documentation

Q. S. Bob TRUONG
Canadian Nuclear Safety Commission, Ottawa, Canada

N. HERBER
Eton Systems, Ottawa, Canada

## Abstract

As part of its mandate, the Canadian Safeguards Support Program (CSSP) is responsible for providing documentation and training assistance to the IAEA. These “knowledge transfer” activities often follow the introduction or enhancement of safeguards equipment and procedures. From time-to-time, an ad-hoc request or sudden procedure change requires the rapid production of new materials. In this paper, we describe some of the lesser-known tools we use to rapidly produce high-quality training documents.

Many safeguards systems are software-based. Measurement and surveillance instruments are controlled by software programs or have data collected from them by software that runs on commercially available hardware. Preparing comprehensive and easy-to-follow training and documentation materials for these systems is a challenge. In two previous papers presented at INMM Annual Meetings, we advocated “guerilla knowledge gathering” and “guerilla knowledge sharing” – using low-cost, low-tech methods to gather and share valuable corporate data. We now focus on “guerilla production” – using simple, low-cost tools and techniques to produce documentation and training packages. We also describe how to set up a stable and repeatable test environment for running new safeguards software.

Although many of these tools and techniques are used by professional writers and producers, anyone who is ever called upon to make a PowerPoint presentation will find tips that will make their next presentation stand out. The tools are easier to use than email, and are of value to anyone doing software documentation for any purpose. Before-and-after examples of simple changes show how to make training materials more informative and lively. A series of “rules of thumb” will help you decide when (or if) to use different types of graphic enhancement – recognizing that the objective is greater reader engagement and better retention of the material.

## 1 Introduction

The Canadian Safeguards Support Program (CSSP) provides support, documentation, and training assistance to the IAEA for Canadian manufactured safeguards equipment and related software tools.

In two previous papers presented at INMM Annual Meetings [Ref. 1, 2], we advocated "guerilla knowledge gathering" and "guerilla knowledge sharing" – using low-cost, low-tech methods to gather and share valuable corporate data. We now focus our attention on "guerilla production" – showing some simple, low-cost tools and techniques that we use when producing documentation and training packages. This includes several methods for setting up a stable and repeatable computing environment to run new software.

Although these tools and techniques can be used with any operating system, this paper only discusses Microsoft Windows. To ensure that these tips and techniques are of use to the widest audience, the examples show Microsoft Word as the publication package, although the tips are applicable to any publication package.

## 2 The motivation

New software-based systems may exhibit many undesirable technical problems and errors – otherwise known as bugs. Running new and untested software on a desktop or laptop machine used for other purposes can be dangerous. Some bugs can damage existing files or otherwise corrupt the operating system.

Knowledge workers charged with the task of documenting a new system need to be able to run the software without endangering their own files. In particular, road warriors who are traveling on business need to be able to test software on their laptop while ensuring that they can still use the laptop for normal business.

In a lab setting, there are usually dedicated test systems set aside to run new software. This is not the case in many situations, such as when the system is remotely located. The tips and techniques described in this paper will be of use to anyone who is working under such circumstances. It is assumed that in the interest of getting the job done, anyone using these techniques is willing to roll up their sleeves and look at a few more technical details than they might otherwise.

To address these problems we have tried several strategies involving both hardware and software. The strategies all have the same objectives:

- isolate the software under test to protect the documenter's machine
- provide a clean computing environment to minimize interaction between the software under test and existing software
- provide a method of recovering from software failures
- simplify the software documentation procedure

## 3 Creating a stable test bed

To mitigate the potential damage caused by new software, we have tried several strategies for creating a stable, isolated, and repeatable test bed. The strategies involve both hardware and software components.

### *3.1 Hardware solutions*

One of the simplest ways to protect the contents of your desktop or laptop machine is to have a dedicated test machine for running new software. A low-cost, generic desktop machine with a minimal number of hardware bells and whistles can be used for testing.

To establish a clean computing environment or baseline system, format and partition the hard drive and install the operating system. Perform any operating system updates required to bring the system up to current standards.

Another invaluable hardware solution is hard disk drive racks. Racks allow the hard drive in a system to be changed (swapped) with the simple flick of a switch and removal and insertion of a disk drawer. Disk drive racks isolate different software systems under test. One dedicated drive can be assigned to each software system or even to different versions of the same software. Hard disk drive racks allow a single computer to be used as both the test bed and as the documentation system.

### *3.2 Software solutions*

Even a dedicated computer or a system with hard drive racks, does not help if rogue software overwrites portions of the hard drive. This kind of protection is provided by drive imaging software. Drive imaging software can take a "snapshot" or a bit-for-bit image of a hard drive and place it on another storage medium. If the software under test should corrupt the hard drive, it can simply be restored from one of the images. The best strategy is to take an image immediately prior to making any major change, such as installing the software under test.

**Windows System Restore** (http://support.microsoft.com/kb/306084/) is not designed to produce bit-for-bit disk images and so is not suitable for this application. It monitors and restores only a core set of specified system and application file types. It is only available on Windows Millennium (Me) and Windows XP. Lastly, the data contained in System Restore's restore points is available for only a limited period of time, after which it is deleted.

**Retrospect**, by EMC Insignia (http://www.emcinsignia.com/), is backup software ($130 US and up) that can make an image of a drive and store it in a file. Compared to the other solutions, Retrospect is slow. Retrospect is also unable to back up some types of open files. Although it is an excellent tool for automated daily backups, we don't recommend Retrospect for taking images of hard drives in a software-testing environment.

**Norton Ghost**, by Symantec (http://www.symantec.com/), has long set the standard for disk imaging. There are two versions of Norton ghost available, Norton Ghost 10.0 ($70 US) for Windows XP or 2000 and Norton Ghost 2003 for other Windows operating systems. Ghost 2003 requires restarting the computer in DOS in order to make the image. Ghost can compress images that it stores, but is still slower than our preferred solution.

**Drive SnapShot** produced by Tom Ehlert Software (http://www.drivesnapshot.de/en/) is our preferred disk imaging solution. Drive Snapshot ($50 US and up) has the unique ability of being able to create disk images on-the-fly while other processes are still running on the computer. It makes a bit-for-bit copy of all files, including those that are open. It is compatible with all Windows file systems and RAID methods and works with Windows NT, 2000, XP, 2003, and x64. It is also amazingly fast, with a save and restore speed of 1.5 GB per minute on a fast processor.

In our test and documentation set up, we use a dedicated computer with two hard drives in racks. One of the racked drives contains the operating system and the software under test. The second racked drive is reserved exclusively for snapshot images. Our test configuration can typically save or restore a complete disk image in two to four minutes.

### *3.3 Virtualization*

Another extremely useful tool for software documenters is system virtualization software. Virtualization software creates a “virtual machine” that runs “inside” the current operating system of the host computer. The virtual machine is completely encapsulated. It cannot affect or damage any of the other programs running on the host computer. Virtualization software creates a disk image of the virtual machine that is stored as a file on the host computer’s hard drive. To create additional copies of the virtual machine, simply replicate that file.

There are three well-known system virtualization software packages available:

- Microsoft Virtual PC (http://www.microsoft.com/windows/virtualpc/) for $129 US
- VMware Workstation (http://www.vmware.com/products/ws/) for $189 US
- Parallels Workstation (http://www.parallels.com/) for $49 US

Virtualization provides several advantages. It does not require any additional or separate hardware (this makes it an excellent choice for use on laptops). It allows you to run both the software under test and your documentation software on the same machine. Any failures or corruption caused by the software under test only affect the virtual machine. Some virtualization software enables many different operating systems to run simultaneously.

The major disadvantage of virtualization is that the software under test is operating on a virtual machine and not on the real hardware that is its target. If the software is controlling or sampling instrumentation, virtualization may not work due to a lack of hardware interface. Virtualization is best suited to pure software applications.

Although we have used virtualization software on some projects, a hardware solution is easier to isolate and manage, and is unlikely to run into problems caused by virtualization.

## 4 Getting great screenshots

To reword an old proverb slightly, “a screenshot is worth a thousand words”. In the case of software-based instrumentation, high quality screenshots can greatly improve documentation and training materials.

### *4.1 Basic screenshots*

There are many different programs available for capturing screenshots. However, only two are discussed in this paper – the Windows print screen utility and TechSmith's SnagIt screen capture software (http://www.techsmith.com/snagit.asp).

The **Windows print screen utility** is part of the operating system. Press the "Print Screen" button (usually near the upper right corner of the keyboard) and a bitmap image of the screen is saved to the clipboard. For an occasional screenshot to paste it into a PowerPoint presentation, the Windows print screen utility is perfectly fine. However, for documenters who need to produce dozens of screenshots, its limitations soon become apparent. It captures the entire screen or an entire window on a single monitor, it captures screenshots as .bmp files, and you need to use another application (PaintShop Pro, for example) to edit, annotate, or even save your screenshots.

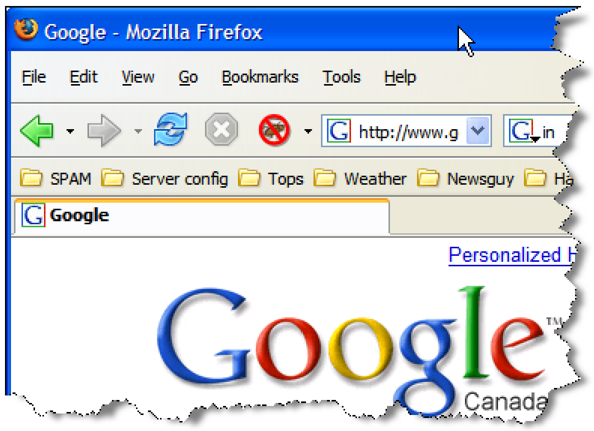
TechSmith's **SnagIt** screen capture software is not free ($39 US), but it overcomes all the limitations of the Windows print screen utility and provides a host of additional features.
It captures any portion of the screen and works with multiple monitors. It saves screenshots in many different formats. It includes an integrated editor that allows you to crop, highlight, annotate, and save your screenshots. It can also capture text from dialog boxes as text (not as a graphic image) and create video captures to show actions on a screen.

There are many different image formats for screenshots. The most common are:

- .bmp – not recommended for documentation because the files are very large.
- .jpg – not recommended for screenshots because it often distorts letters, lines, and other small screen details.
- .gif – recommended because it is compatible with many programs. Screenshots captured in .gif format can produce very small files with good fidelity.
- .png – recommended because it does an excellent job on screenshots and is compatible with all browsers, Microsoft Word, and Microsoft PowerPoint.

### *4.2 Advanced screenshots*

Despite their name, screenshots rarely need to show the entire screen. It is more useful to show only one of the windows on the screen or a portion of a window. The rule of thumb is: "if you don't need to see it, don't show it."

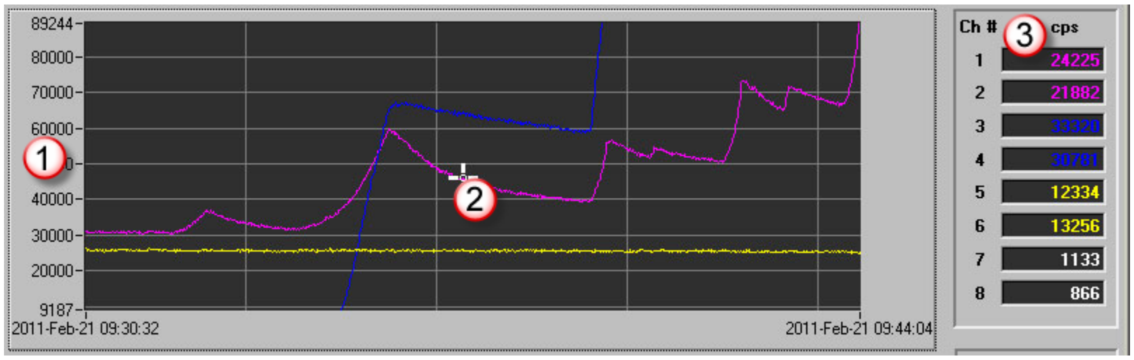

Google - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.g
SPAM Server config Tops Weather Newsguy
Google
Personalized
Google
Canada

To help readers establish the relative position of a partial screen, use an effect such as a ragged edge on the image. As shown in this figure, it looks as though the image were torn from a page. The ragged edges immediately tell the reader where on the screen the items are located.

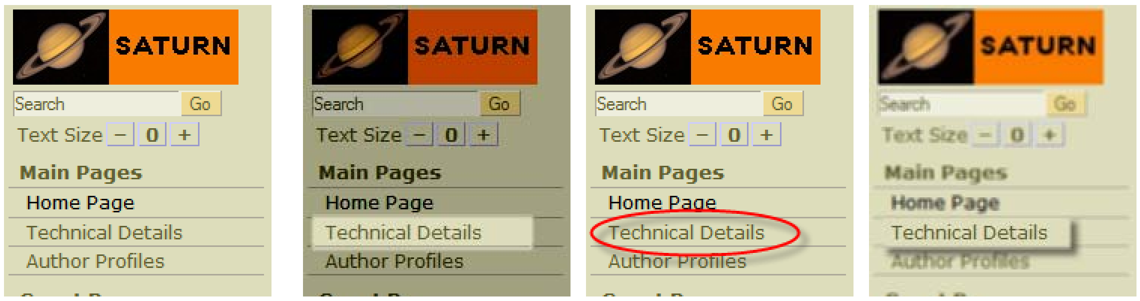
It is important to keep approximately the same scale for different screenshots within one document. For example, if a dialog box is shown at 75% of its normal size, the same scale should be used when showing a menu. One an exception to this rule is when showing an entire screen. It may need to be reduced to 50% of normal size to fit on a page.

Leave the original screenshot in its native resolution. Any scaling should be done in the publication package (or on a copy of the original). This allows the original screenshot to be reused for other purposes where the scale may differ.

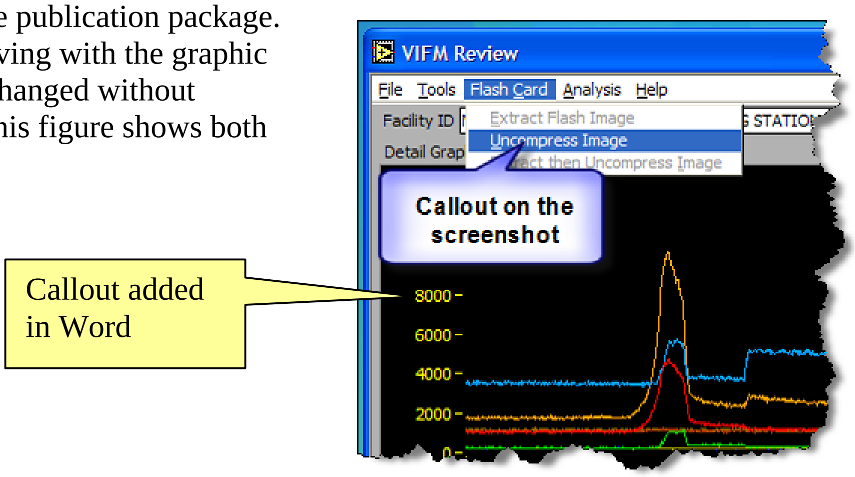
When describing various regions (buttons, data boxes, and so on) on a screen use the SnagIt stamp tool to note the areas of interest with numbers. The figure below shows an example of enumeration.



Highlighting tools focus attention on the objects of interest. The figures below show an original screen segment with three different examples of highlighting.



Rather than describing items on the screen elaborate detail, use “callouts” to both indicate the item and describe it. Callouts can be added two ways: as a graphical stamp (part of the screenshot), or as a separate callout added in the publication package. Stamped call outs always moving with the graphic while added callouts can be changed without recapturing the screenshot. This figure shows both types of callouts.



## 5 Practicing what we preach

In order to illustrate the application of the techniques and tips discussed in this paper, here are some real-world examples of our guerilla production experiences from three recent projects at the CSSP.

**Example 1 - Content reuse:** Several years ago, the CSSP produced a training CD for the IAEA for a spent fuel monitoring system known as VIFM. The CD included several step-by-step procedures for inspectors to follow when servicing the equipment. We wanted to reuse some of this material as part of a PowerPoint presentation in a training course. Unfortunately, the source material was in a format incompatible with PowerPoint. We solved this problem by using SnagIt to capture a video of a fixed region on the screen while playing the CD. The captured video was used directly in PowerPoint.

**Example 2 - One-click install test: –** The CSSP provided the IAEA with VITA, a text visualization and analysis tool, which allows knowledge workers to see a visual representation of a search on the Web or on a local collection of documents. VITA makes use of a variety plug-ins supplied by third parties. It was important to ensure that the one-click install procedure would work on various operating systems with all of the plug-ins. Drive SnapShot was used to set up several clean environments using different versions of Windows. We were able to test the installer on these different versions, report any problems to the developer, and revert to the original clean environment, ready to test the next version. SnagIt was used to capture screenshots of the installation procedure and the operation of the software itself for documentation and training manuals.

**Example 3 - Using software stubs:** One of the problems we faced early on was documenting the VIFM spent fuel monitoring system, an IAEA safeguards instrument used at nuclear generating stations. Access to the VIFM system was limited, and because it ran under a locked-down operating system it was impossible to install utilities such as SnagIt. Initially we could only use the Windows print screen utility. We needed screen captures of the error messages the system produced, and it was laborious (and in some cases, impossible) to create all of the error conditions that the system could detect.

The solution for this problem was to create a special “stubbed” version of the software. This version runs on any PC and has extra software components known as “stubs”. Stubs mimic the operation instruments and detectors attached to the system and supply well-controlled, simulated data to the system. In effect, the stubs turn a host PC into a virtualization of the full instrument.

The stubs are configured in such a way as to simulate all the possible error conditions that the system can detect. This allowed us to generate all the screenshots we needed while running the software on a laptop. Because the stubbed version was not dependent upon particular hardware or a locked-down operating system, we were able to use SnagIt for screen captures. As a side benefit, the stubbed software is also available for inspectors to use for servicing rehearsal exercises.

## 6 Concluding Remarks

This paper was originally intended as a knowledge-sharing instrument for some of the screenshot tips and tricks we had learned while producing manuals, documents, and courseware for the IAEA. However, it soon became evident to us that a far more important, and less understood, topic was creating an isolated, robust, repeatable computing environment (a stable test bed) in which to run the software we were documenting. This was particularly evident on the VITA project (example 2, above) where we used SnagIt extensively to document the package, but we needed to reinstall the software several times to test the one-click install procedure. Snapshot disk images of the baseline systems allowed us to quickly revert the test machine back to a known state.

Equipment development is an important and substantive item in the IAEA Safeguards Department and many safeguards systems are software-driven. We hope that the techniques and experiences discussed here will prove useful to IAEA staff and their consultants when testing and documenting software for safeguards equipment.

We invite anyone interested in further information about these topics to visit our wiki [Ref. 2, 3] at http://saturn.eton.ca/. It features live web links, contact information, and expanded discussion of this paper.

## ACKNOWLEDGEMENTS

The authors wish to thank the following for their advice and assistance in preparing this paper: Ken Desson, Androcom Interactive Media, Ottawa, Canada; Cesare Liguori, International Atomic Energy Agency, Vienna, Austria

## REFERENCES

1) Truong, Q.S. (Bob); Keeffe, R.; Herber, N.; Barroso, H.; *Just-In-Time Knowledge Capture Techniques*, paper presented at the INMM 45th Annual Meeting, Orlando, FL, USA, July 18-22, 2004.
2) Truong, Q.S. (Bob); Herber, N.; Barroso, H.; Liguori, C.; *Using Wikis as a Low-Cost Knowledge Sharing Tool*, paper presented at the INMM 46th Annual Meeting, Phoenix, AZ, USA, July 10-14, 2005.
3) A WikiWikiWeb site with live web links, contact information, and expanded discussion of this paper: http://saturn.eton.ca/ .