# SYSTEM CONVERSION GUIDELINES for NEW OPERATING SYSTEMS & SOFTWARE

These guidelines are really about **good business planning**. Speaking from decades in business consulting & management, and computer training & support (and countless computer system conversions), I've learned **six basic rules** for operating system and applications software changes, which may help you.

**1. MAKE SURE THE CURRENT HARDWARE IS UP TO THE NEW SOFTWARE** (adequate storage, memory, processor speed, and other features). **In most cases, a software 'upgrade' actually *down*grades computer performance**, and requires (really REQUIRES) a corresponding hardware upgrade -- sometimes to maintain the same level of performance, sometimes to make the new software work at all.

**2. CHANGE ONE (1) THING AT A TIME** -- if at all possible.
**If not possible, change as FEW things as possible**.
If a <u>*complete*</u> replacement of hardware, operating system and applications software is the ultimate objective, just
**DO IT IN STAGES**. This is critical for TWO reasons:

**a.)** If a problem crops up (and it ALWAYS does), it's a lot easier to find out the cause -- and solution -- if only **one** variable has changed. An old rule of computer troubleshooting is that the complexity involved in tracing and resolving a problem varies *exponentially* -- by the square (or one-half of the square) of the number of variables involved. In other words, changing 3 things (rather than 1) results in 4-and-a-half times the trouble -- or 9 times the trouble. If a whole lot of things change at once (and I've seen this happen to several organizations, and been called in to help them with the resulting disaster) it may never be possible to untangle the mess without simply pulling out everything, and starting over... by changing ONE thing at a time.

**b.)** Even if there are NO hardware or software problems, changing *everything* all at once puts an enormous workload on the computer **users**, because of the need to learn multiple new programs (and possibly even new hardware items/functions). Hitting a work crew with a new operating system – especially one full of new functions and new user interfaces and presentation styles (like *Windows 10*, or switching to Apple) creates a massive learning curve that disrupts productivity and efficiency.

To compound that – with a requirement to learn a new word processor, new presentation software, new accounting software, new e-mail, new web-browser, new drawing software, new image processing software, etc. – can create a vertical learning curve that brings productivity to a standstill. During the transition, it doesn't generally help for management to push harder on the employees to keep productivity up to pace with normal (pre-conversion) performance. (Sometimes good employees have a nervous breakdown and/or quit, as a result.)

**3. HAVE A BACKUP AND REVERSION PLAN, AND BE READY TO USE IT. *Never* assume the change is going to work – at least not the *first* time.** An organization ***must*** have accessible backups of EVERYTHING (data AND software), and be prepared to back up and start over, in case of a major foul-up. **Things DO happen.** A new desktop computer *operating system* may run afoul of a certain computer/network/server/server-operating-system combination, or snag certain printers/printer-drivers (or other necessary hardware and drivers), or refuse to work with certain necessary legacy software.
New *applications* software may run afoul of those same things – and may also (and often does) fail to work as advertised with the new operating system – especially if the hardware isn't the latest, greatest, and most-standard stuff possible.

If things get out-of-hand, a properly prepared set of backups (and management and technical support) *should* permit a **complete** reversion to the "*status quo ante*" (the way things were before) -- allowing everyone to get back to work (and enabling productivity to resume) while the computer guys work out the problems with the new stuff.

**4. TEST BEFOREHAND.** (This guideline, and the next one, actually should be listed as guidelines #1 and #2, but no one appreciates them until the other steps are explained and considered.) ***Before*** changing a *worker's* computer, the computer guys should get a *test* computer (configured like a *normal* worker's computer – **not** a fancier or plainer system), and set up a few simple tests of the new operating system in the different hardware/ software / environment configurations expected, and test to see if *everything* works.

Check that all app software works with all OS software on current hardware. Ensure every application can open, save, preview, print (with preview results matching actual output), cut-and-paste *within* the application, cut-and-paste between *different* applications, and interact with all normal input, output and storage devices. Ensure fonts carry across. Amazing how many problems can be discovered in advance, and headed off by this technique. Much more work for computer staff, but much less for everyone else, for lower overall business cost.

**5. IMPLEMENT IN STAGES.** Have the most-proficient computer users tackle the first upgrades. Then, when *they* are proficient with the new system, have *others* follow – while using those early 'experts' as guides for the other workers. Plan and budget consulting/training time for that 'software guru' to get up-to-speed – and to train and assist others. The *long*-term NET effect is usually a MAJOR net gain – and even the *short*-term consequences will usually justify/recover the expense.

**6. BUDGET FOR THE <u>*WHOLE*</u> COST.** Upgrades of "just this piece of software" actually almost always require budgeting for several more *actual* ultimate costs:
**a.) The DESIRED software upgrade** (**parts *AND* labor**).
**b.) OTHER software upgrades**, made necessary by the desired software change (both parts & labor).
**c.) Hardware upgrades**, as necessary (parts & labor).
**d.) Computer technical support** (often the server/network-support people get involved, or printer vendors, etc.)
**e.) Downtime for in-house "guru"** - sidetracked from his/her regular work to master the new system.
**f.) Training for staff** (including: staff time, costs of having staff unavailable for normal work, cost of trainer time, costs of having trainer unavailable for normal work, and costs for downtime of hardware temporarily diverted to training).
**g.) Temporary productivity loss**. **Nearly ALL significant system upgrades result (initially) in a LOSS of worker efficiency/productivity** while the bugs get worked out and people learn all the new stuff. This is generally the least expected (and usually, by far, the most costly) consequence of an 'upgrade.' It's important to plan for the *full* duration and cost of a realistic learning / troubleshooting / adaptation curve.

---

**If these guidelines are followed, most organizations that use them have pretty good, fairly predictable results**, and the upgrades are seldom a disaster. Organizations that don't follow this model, commonly have very opposite results – often not recognized until the annual financial report. *~RH*