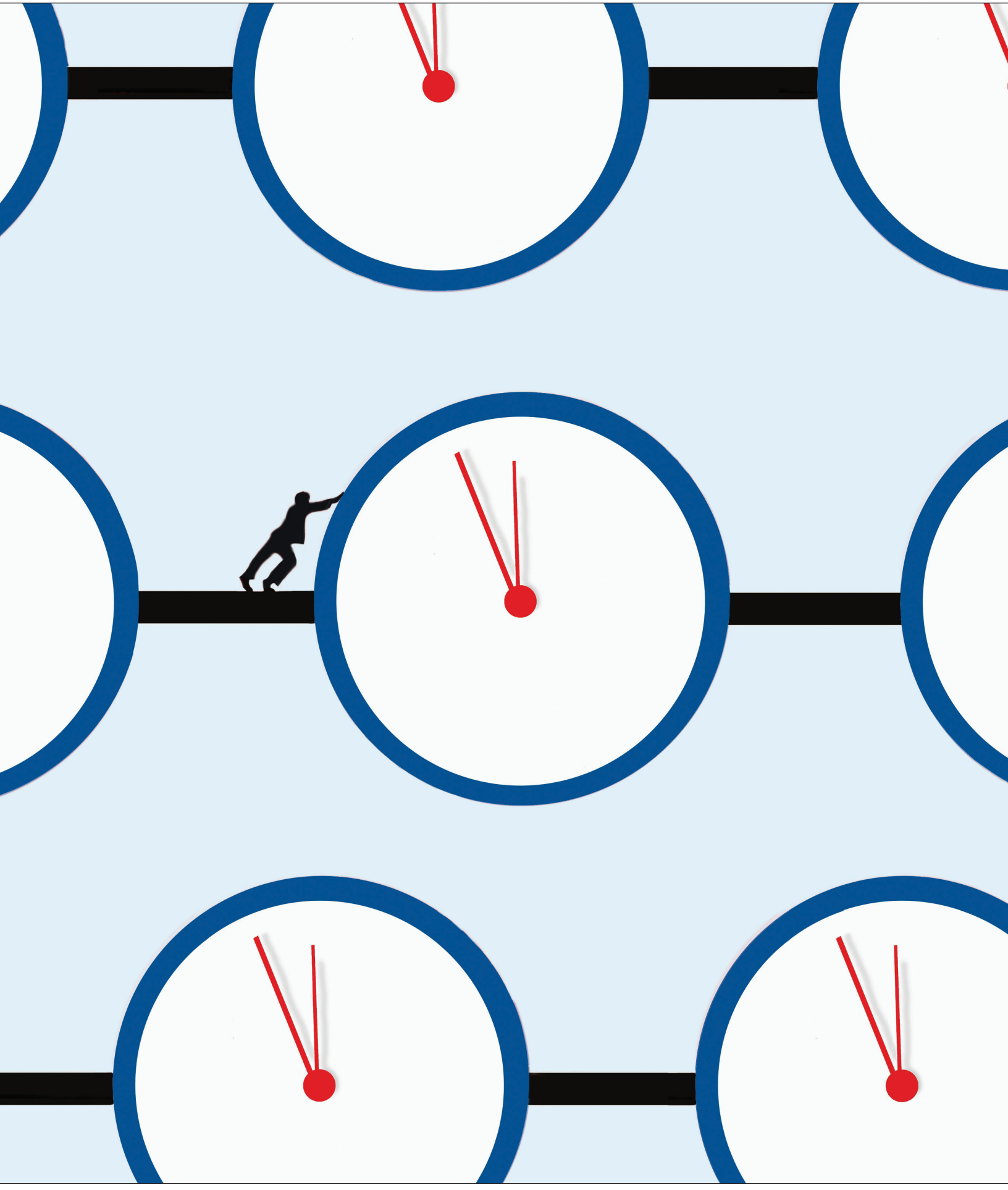


RETHINKING BUDGETING

Rework

Doing Double for Nothing and How to Fix It

BY SHAYNE KAVANAGH

About GFOA's Rethinking Budgeting Initiative

Local governments have long relied on incremental line-item budgeting, in which last year's budget becomes next year's with changes around the margins. In a world defined by uncertainty, this form of budgeting puts local governments at a disadvantage, hampering their ability to adapt to changing circumstances.

As we all know so well, the ability to adapt has become essential over the last two years—and will certainly remain so for some time. The premise of the Rethinking Budgeting initiative is that the public finance profession has an opportunity to update local government budgeting practices with new ways of thinking and new technologies to help communities better meet changing needs and circumstances. The Rethinking Budgeting initiative seeks out and shares unconventional but promising methods for local governments to improve how they budget, and how they embrace the defining issues of our time.

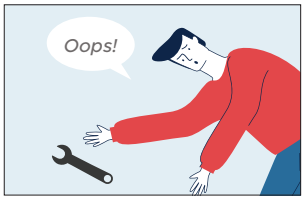
This issue of *GFR* looks at three key subjects underneath the umbrella of Rethinking Budgeting: rework, negotiation and persuasion, and equity and equality. The following three articles provide current perspectives as well as guidance on how local government finance professionals can put them into practice.

Rework is when you have to do a task over again because it was not done right the first time. This could add up to double the time needed to complete a task! Rework is a big problem for public finance officers. It ranked as the third largest source of wasted time in a poll conducted by GFOA.[1] In this article, we will explore how to prevent rework.

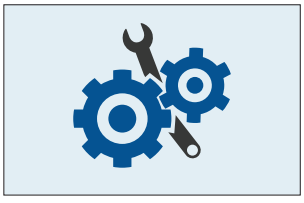
We can learn a lot about how to prevent rework from "Lean"[2] process improvement. Lean has been used successfully for decades in many industries, including government, to reduce waste in the workplace, including waste from rework. Lean teaches that preventing rework requires getting to the root cause of rework. According to Lean, the immediate cause of rework is "defects." A defect happens when incorrect or incomplete work is sent to the next step in the process or to the customer. Our goal should be to perform work right the first time. This can be achieved by designing and controlling the work process to deliver work that is free from defects, thereby eliminating rework. In this article, we will help you design your work process to reduce the probability of defects.

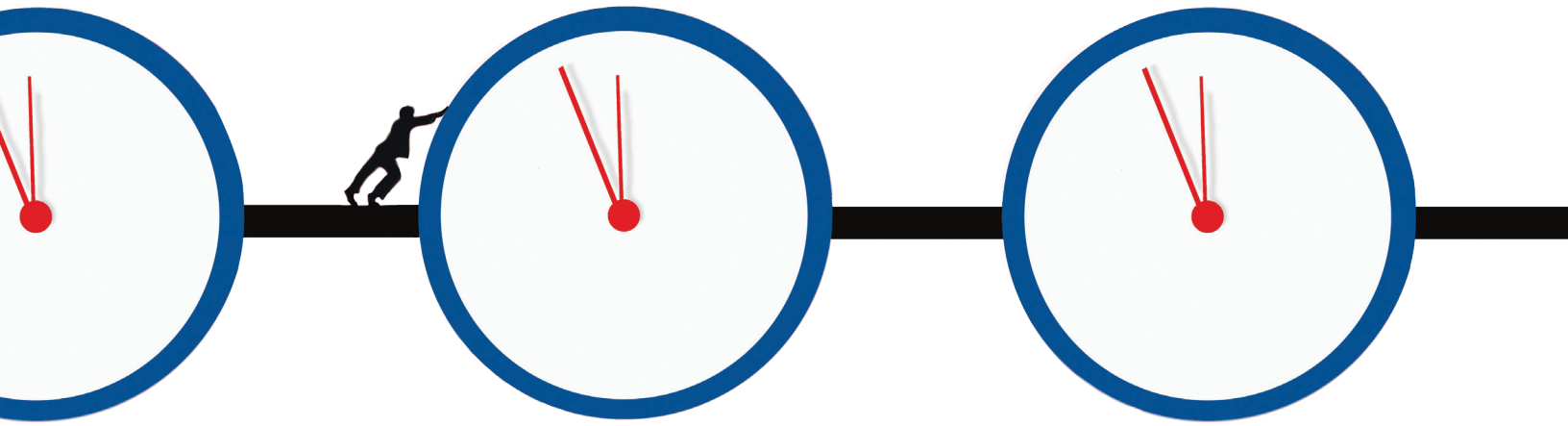
ERRORS VS. DEFECTS

An error only becomes a "defect" when the faulty work is passed on to the next step in the process and becomes a wrench in the works.



ERROR


DEFECT



Let's start by defining our core principles for reducing defects:

1. **You cannot inspect your way to quality.** Inspection or "quality assurance" is not a good solution to defects because our goal is for work to be done right the first time. According to Lean, inspections are a waste because they cost more than doing the work right the first time. Inspections add time to a task and any mistakes caught have to be corrected, which is still rework.
2. **To err is human, but defects are avoidable.** People are not perfect. They will sometimes commit an error. This is unavoidable. However, an error only becomes a "defect" when the faulty work is passed on to the next step in the process and becomes a wrench in the works. Our goal is not to eliminate human error (which will never happen) but prevent errors from becoming defects by providing feedback and action at the point of an error. For example, if the gallons of water consumed by a utility account are entered incorrectly into the billing system, that is an error. It becomes a defect when an incorrect water bill is sent to the customer.
3. **We can design error- or mistake-proofing mechanisms.** Systems can be designed to reduce the probability of a defect. Error-proofing mechanisms can either: A) detect the error when it occurs and alert a person who can fix it or B) prevent the error from becoming a defect. To continue our utility billing example, a preventative error-proofing mechanism is automated meter reading that eliminates the need for a person to type the gallons consumed into a billing system. A detection mechanism produces an alert when the gallons consumed are out of line with historical consumption patterns (as might be the case with detecting potential leaks, for example).

So how can we design mistake-proof mechanisms? The first step is to describe the defect and collect data on it. Where is the defect discovered? Where does the error occur that creates the defect? How common is this defect?

Next, observe the process and compare it to your standard operating procedures (SOPs). Are there errors in the SOPs or are workers deviating from the SOPs? If you don't have clear SOPs, we probably just identified a source of errors and defects.

The third is to analyze the root cause for each error or deviation from the SOPs. Critical to doing this well is to adopt the mindset that though people make mistakes, people are typically not the root cause. For example, concluding that people are careless or dumb as the root cause is a dead end for error proofing. If you conclude that people are the root cause, the only solution is often replacing the people, which might be impractical or costly. Instead, your goal should be to figure out how the work process can be designed to succeed despite the errors people make.

> Your goal should be to figure out **how the work process can be designed to succeed** despite the errors people make.

To illustrate, below are common errors people make and potential problems in a work process that can cause these errors:

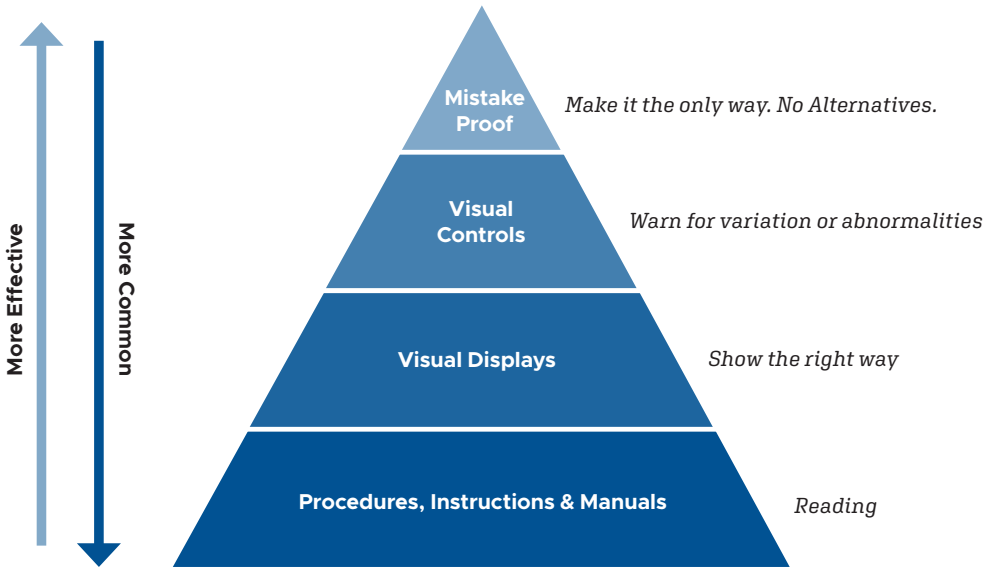
- **Distraction.** Interruptions are a widespread and pernicious cause of distraction and lost productivity.
- **Forgetfulness.** Distraction could also be a cause of forgetfulness. Another cause that is germane to the finance office might be that, for some participants in a process, the process may be a low-priority task. For example, people in other departments likely regard tasks like filling out forms for time entry, purchasing, etc., as necessary evils rather than core job functions. Hence, these tasks fall to the bottom of their to-do lists. So the solution might be to produce better reminders or streamline the task so that participants are more likely to complete it.
- **Lack of experience.** Here the root cause might be inadequate training or ineffective instructions.
- **Misunderstanding instructions.** This could arise from poor instructions. This could be due to poor writing, overreliance on text (not enough pictures), or the use of jargon that is known to the authors of the instructions but not to the user.
- **Lack of standards.** If there are no written SOPs, the process will not be performed consistently. Variation in how a process is performed is a sure source of errors.
- **Not following SOPs.** If people are not following SOPs, are they aware they exist? Are the SOPs clear? Has management emphasized the importance of the SOPs? Perhaps most important, have participants in the process been consulted in the development of the SOPs? Deviations from the SOP might result from some issue that continually comes up in the course of performing the work, which the SOPs do not adequately address. So the workers ad-lib their response to that issue. Also, people will be more likely to follow the SOPs if they had a hand in developing them.

The last step in the design is to generate ideas to eliminate the error or detect the error early, create the error-proofing device, and then test, validate, and implement the error-proofing device. The rest of this article will be dedicated to helping you design error-proofing devices.

What does error proofing look like? Let's start with general features. Ideally, error-proofing devices are inexpensive and simple. They do not require you to buy a whole new technology system, for example. Error-proofing devices should give prompt feedback to the process participant that an error has occurred and/or initiate preventative action to stop an error from becoming a defect. The error-proofing action should occur before the error goes on to become a defect. Finally, error-proofing devices should have input from people who do the work. They may have ideas to improve the device or prevent the error and will be more likely to work with (rather than against) the error-proofing device if they have given input.

Now let's get specific with a tool called the "compliance pyramid," shown in Exhibit 1.[3] This helps you think through

EXHIBIT 1 | COMPLIANCE PYRAMID[3]



Though a comprehensive procedures manual is useful, **it may not be suited for everyday use.**

your error-proofing options. At the bottom of the pyramid, we have written instructions. These are the most common types of error proofing but also the least effective because they require reading. The next level up is visual displays. This is putting instructions into visual form. This could include pictures of how a process is performed or markers (like colored arrows) that guide a user from one step to the next. Going further up the pyramid, we arrive at visual controls. These warn the user that the action performed has fallen outside of normal parameters. Earlier, we gave an example of a water billing system that checks a customer's water usage for deviation from historical patterns. This might produce a warning to the utility department or customer in the form of an email. Finally, at the top of the pyramid is to make a step mistake-proof. Earlier, we gave the example of automating meter reads to eliminate human intervention.

What opportunities for mistake proofing do finance offices have at each level of the pyramid? We'll offer some ideas here, but we encourage you and your colleagues to think creatively about how to redesign the work environment to reduce or eliminate errors.

Let's start with written instructions, since those are the most common. They are foundational, as the compliance pyramid implies: All other controls will be based on the SOPs. However, many governments don't have written SOPs for things like year-end close, budget requests, or grant reporting. Steps are just learned and passed down from one employee to the next. When SOPs don't exist, procedures need to be "relearned" when there is turnover.

Some governments have developed comprehensive procedures. For example, the California Courts developed a policies and procedures manual that covers the following topics:

- **Budgets**
- **Accounting Practices**
- **Procurement**
- **Contracts**
- **Accounts Payable**
- **Fixed Assets**
- **Collections**
- **Audits**
- **Record Retention**
- **Banking and Treasury**
- **Security**
- **Fund Accounting**

Though a comprehensive procedures manual is useful, it may not be suited for everyday use. Hence, there may be opportunities to create streamlined written controls that focus on the steps a user needs to know. The GFOA Ethics Policy templates (available at www.gfoa.org/materials/templates-for-everyday-ethical-challenges for GFOA members only) provide an example. Some features of these templates that make them more accessible include:

Written in a conversational tone. Policies are often written in a formalized language. This can make the policy difficult for people to relate to.[4]

Keep the language simple. The average American reads at a seventh- to eighth-grade level. A policy written at about this level will be widely understood.[5] Microsoft Word can score the readability of your document using two common measures: the Flesch-Kincaid Grade Level test and the Flesch Reading Ease test.[6]

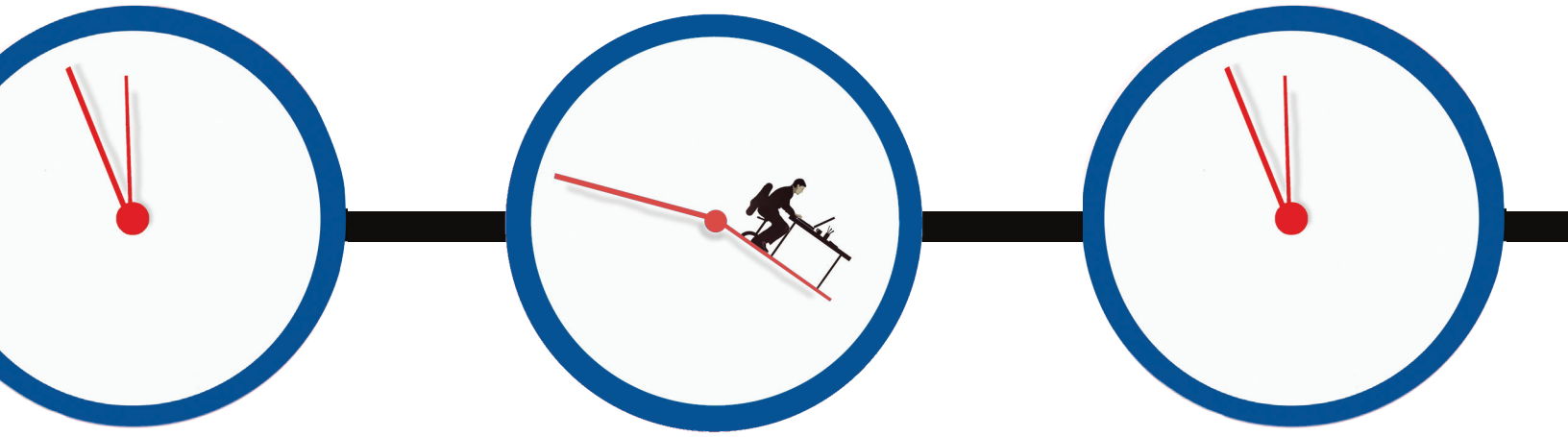
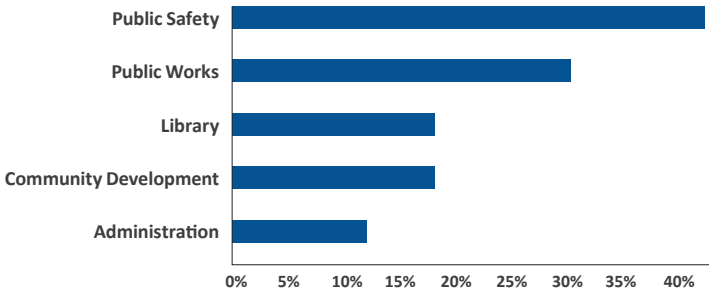


EXHIBIT 2 | A CHECK CELL IN EXCEL

Percent of Budget	
Administration	10%
Community Development	15%
Library	15%
Public Works	25%
Public Safety	35%
TOTAL	**100%**



Keep the core policy simple. If the procedures can be short and to the point, there is a chance people will remember them.[7] The GFOA Ethics Policies templates present a series of essential "need to know" bullet points. But for a control device, a better approach is a checklist. Simple checklists have proven powerful for improving adherence to potentially life-saving procedures in fields as diverse as aviation and healthcare. If checklists can save lives, they can save the time and money associated with rework. A good checklist does not reproduce every step of the procedure, rather it focuses on the critical steps that are likely to be overlooked. *The Checklist Manifesto*[8] provides an excellent guide to the value of checklists and how to create them.

Written SOPs can also provide the basis for visual aids that show the right way. A starting point would be to include screenshots with written instructions to show how to interact with computerized information systems. There are also other ways. Pasco County Schools created videos to show how to process transactions (available at www.pasco.k12.fl.us/otis/page/munis) as an alternative to conventional documents.

Visual aids are not limited to a passive explanation of a process; they can also be designed to actively guide the user through the work. For instance, forms often ask the user for more information than is needed to process the transaction. Removing these fields declutters the form, makes it easier to follow and complete, and focuses the user on the most important information. Forms could also be visually designed to emphasize the work flow. For example, if there is a sequence that is important to follow, then the fields could be numbered or arrows used to guide the user's attention from one part of the form to the next.

A more active aid is the "wizards" that are built into some of the enterprise resource planning (ERP) software that many local governments use. Wizards walk employees through processes like onboarding or benefit open enrollment questions. The wizards ask questions (for example, Do you have dependents?) and route the employee to the appropriate next step (as in, list dependents, provide proof of eligibility for benefits). That said, wizards are not available in all ERP systems, and the cost of implementing and maintain the technology is probably only justifiable for larger organizations.

Technological visual aids don't need to be as sophisticated as a wizard. For example, an Excel form could color code the fields that the end user is expected to fill out.

Visual aids show the right way. The next step up the compliance pyramid is visual controls, which warn the user if a mistake occurs. An example is when required fields turn red if a user attempts to proceed without completing those fields. Excel offers many opportunities to use visual controls. For instance, Excel forms can use conditional formatting to highlight values that are outside of parameters. Another strategy is to create a "check cell" that will contain a given value if a series of other cells compile to the correct value. A simple illustration appears in Exhibit 2. The user wants to develop a chart that shows the percent of the budget consumed by each department. The check cell appears in black. It shows that the numbers add up to 100%, but that cell is not used in the end product (the graph).

ERP systems can provide visual controls as well. For example, required fields can be color coded. Similar to a check cell, some ERP systems will check the net total on a journal entry. The running total will indicate whether or not the entry is balanced.

The pinnacle of the compliance pyramid is to mistake proof the process, so the right way is the only way. Let's start with some simple illustrations in Excel. Data validation rules can be used for input cells. To continue our example from Exhibit 2, the form could require that the numbers entered

for each department are percentages. This example could be extended to other applications. For example, GFOA builds risk models that help local governments analyze the vulnerability of their reserves, recessions and extreme events. The model gives the user the option to activate budget-balancing strategies in response to budget stress.

If the user wants to activate a furlough, then the user must specify how many positions to furlough. The model restricts the user to a range of positions. This prevents, for example, the user from simulating a furlough of more positions than exist!

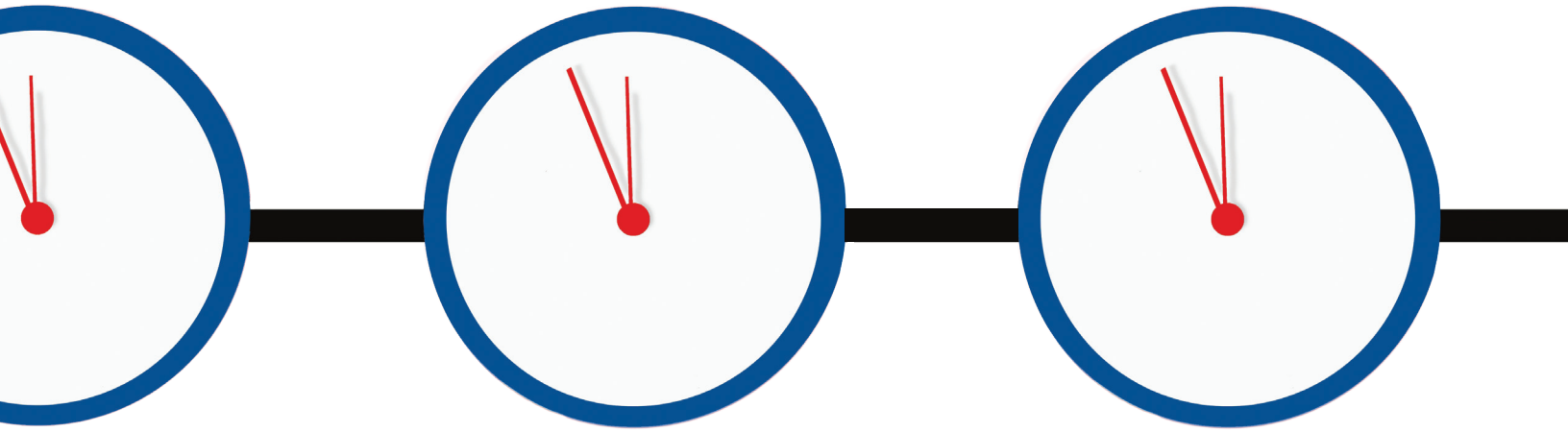
Another mistake-proofing example is to restrict access to cells that perhaps the user would like to see but should not change. This can be done in Excel by locking certain cells.

ERP software has several mistake-proofing possibilities. For example, a purchase requisition form might require uploading quotes before the user can proceed or it might require entering valid commodity codes. ERP systems also can default values. For example, the line items a user has access to can be limited to that user's department. Finally, an ERP system might refuse to post journal entries that are not balanced and notify users that they are entering an unbalanced transaction.

Conclusion

Rework can be a serious time waster because it means you have to do the same job twice. This article has described how you can reduce rework by eliminating the defects that cause it. Once you've identified processes in your workplace that generate a lot of rework, you can use the steps described in this article to reduce the amount—and save more time.

Shayne Kavanagh *is the senior manager of research for GFOA's Research and Consulting Center.*



[1] Poll conducted via GFOA newsletter. The first two time wasters were meetings and interruptions, respectively.

[2] Kavanagh, Shayne; Cole, Jeff; Kenworthy, Harry (January 2014). A guide to starting the lean journey. GFOA. https://www.gfoa.org/materials/a-guide-to-starting-the-lean-journey

[3] Teeuwen, Bert (December 14, 2010). Lean for the public sector: The pursuit of perfection in government services. Productivity Press; 1st edition.

[4] Pinker, Steven (2015). The sense of style: The thinking person's guide to writing in the 21st century. Penguin Books. Pinker, a cognitive psychologist, stresses the importance of a conversational tone in writing.

[5] Using text readability formulas: It has been shown that the average reading level is anywhere from seventh to eighth grade, indicating that a sixth grade level would likely be widely accessible to U.S. adults. See http://www.clearlanguagegroup.com/readability

[6] To use this feature on a Windows computer, go to File > Options; select Proofing; under When correcting spelling and grammar in Word, make sure the Check grammar with spelling checkbox is selected; select Show readability statistics. The readability statistics will be displayed when you do a spelling and grammar review.

[7] Heath, Chip; Heath, Dan (2007). Made to stick: Why some ideas survive and others die. Random House. Note that the authors don't suggest that shorter is necessarily better. Substance is important too.

[8] Gawande, Atul (December 22, 2009). The checklist manifesto: How to get things right. Metropolitan Books. http://atulgawande.com/book/the-checklist-manifesto