

Learning Organizations and the Software Developer

Warren Harrison

I once heard a well-known software development guru define insanity as “doing the same exact thing over and over and expecting different results.” I’m not convinced this is truly the definition of “insanity,” but it certainly fits the term “clueless.” Obviously, the entire point of this guru’s “definition” was to point out that when someone isn’t learning from their mistakes, they can’t expect things to improve.



If an organization fails to learn from its experiences, no matter how successful it might be initially, we can expect its performance to eventually decline to a point where the organization is no longer viable. History is littered with companies that were once the favorites of Wall Street but failed to effectively change and adapt, gradually sinking into obscurity.

On the other hand, a learning organization is continually increasing its capacity to produce results. Most software development organizations strive to become learning organizations, but far too few are successful. Why is this so rare? With some reflection, it’s easy to identify a three-step process any learning organization must master. First, the organization learns something; second, it manages this knowledge; and finally, organizational behavior is somehow changed by virtue of this new knowledge.

Recognizing mistakes and successes

To “learn something,” the organization must be able to recognize its and others’ mistakes and successes. The organization’s eyes and ears are its people. When someone in the organization recognizes a mistake or a success, be it internal or external, he or she must be willing and able to make note of it. However, recognizing mistakes and successes isn’t free. It takes time to reflect on what went right and what went wrong. It takes real effort to abstract an event’s key issues so that they’re applicable elsewhere.

When the successes are from outside the organization, it’s even more difficult and time consuming to keep on top of things. Whereas much of what we know about our organization comes from either our direct experience or that of someone we trust and respect, other organizations’ experiences almost always come from someone we don’t know. They might appear in a journal article or workshop presentation or be relayed by a consultant. Some suspicion toward the messenger and the message is natural.

Academic software engineering researchers are one of the few groups of people actually paid to package, evaluate, and distribute the experience of software development organizations. But this doesn’t necessarily mean they’re trusted by one and all. The burden of proof required from a stranger is much higher than what we expect from a colleague.

DEPARTMENT EDITORS

Bookshelf: Warren Keuffel,
wkeuffel@computer.org

Construction: Andy Hunt and Dave Thomas,
(andy, dave)@pragmaticprogrammer.com

Design: Martin Fowler,
fowler@acm.org

Loyal Opposition: Robert Glass,
rglass@indiana.edu

Manager: Don Reifer,
d.reifer@ieee.org

Quality Time: Nancy Eickelmann,
nancy.eickelmann@motorola.com,
and Jane Hayes, hayes@cs.uky.edu

Requirements: Suzanne Robertson,
suzanne@systemsguild.com

STAFF

Senior Lead Editor
Dale C. Strok
dstrok@computer.org

Group Managing Editor
Crystal Shif

Senior Editors
Shani Murray and Dennis Taylor

Assistant Editor
Rebecca Deuel

Editorial Assistant
Joan Shim

Magazine Assistant
Pauline Hosillos, software@computer.org

Art Director
Toni Van Buskirk

Cover Illustration Technical Illustrator
Dirk Hagner **Alex Torres**

Production Assistant Production Artist
Monette Velasco **Carmen Flores-Garvey**

Executive Director
David Hennage

Publisher Assistant Publisher
Angela Burgess **Dick Price**

Membership/Circulation Marketing Manager
Georgann Carter

Business Development Manager
Sandra Brown

Senior Production Coordinator
Marian Anderson

CONTRIBUTING EDITORS

Robert Glass, Kirk Kroeker, Anne Lear,
Rita Scanlan, Keri Schreiner

Editorial: All submissions are subject to editing for clarity, style, and space. Unless otherwise stated, bylined articles and departments, as well as product and service descriptions, reflect the author's or firm's opinion. Inclusion in *IEEE Software* does not necessarily constitute endorsement by the IEEE or the IEEE Computer Society.

To Submit: Access the IEEE Computer Society's Web-based system, Manuscript Central, at <http://cs-ieee.manuscriptcentral.com/index.html>. Be sure to select the right manuscript type when submitting. Articles must be original and not exceed 5,400 words including figures and tables, which count for 200 words each.

Distributing knowledge

Managing knowledge requires a mechanism for storing and distributing the experiences people capture. Human memory is a fleeting thing. Expecting someone to remember the details of an experience months later is unreasonable. Expecting that person to always be physically present at a time and place where knowledge of this experience is needed is downright delusional. There needs to be a corporate memory that can manage the knowledge contributed by the organization's people.

This must be proactive as well as selective. Expecting workers to poll colleagues for their knowledge—especially in geographically diverse organizations—is unrealistic. However, a constant stream of irrelevant (to the recipient) pieces of knowledge will just about ensure that the relevant knowledge will be missed. Like many things in life, there's a fine balance between too much and too little.

Fortunately, this can be the most straightforward aspect of being a learning organization because it lends itself to a technical solution. For example, the US National Aeronautics and Space Administration has an extensive Lessons Learned Repository at <http://llis.nasa.gov/llis/plls/index.html>. This is a good example of how an organization might choose to disseminate its people's knowledge.

Open minds and empowerment

Even with the first two ingredients, if the organization fails to change and adapt in response, there will be little if any benefit. People must be both willing and empowered to embrace this new knowledge and change their behavior based on it. It does no good to tell workers that a procedure was a great success when someone else carried it out in a certain way if they are forbidden from changing how they do the procedure. Likewise, if workers are closed to new ideas or so risk-averse that they can't chance failure, it's highly unlikely that any new knowledge will affect the way they do business.

These three ingredients result in a *learning-distribution-change* chain. In other words, the organization learns something, it distributes this knowledge, and organizational behavior somehow changes. Missing only a single ingredient can break the chain. If the organization can't learn from its experiences or the experiences of others, no new knowledge exists. If the organization can't maintain and distribute this knowledge, members won't learn about it. If the recipients of this new knowledge aren't willing or able to take it to heart, change won't happen.

Impediments to creating a learning organization

A preliminary analysis of the problem of fostering a learning organization might lead us to believe that simply instituting a lessons-learned program would ensure a learning-distribution-change chain. However, the problem is much more complicated. I recently came across an article in which the author listed impediments to creating a learning organization. I'll paraphrase it here:

- Skepticism about research as ivy tower and impractical.
- Resistance to cooperating with outside researchers because too often they've failed to provide feedback soon enough to assist practitioners.
- Distrust of evaluation research because of the blisters that linger from the last time a badly conducted study burned the organization.
- Skepticism that research findings developed in another organization have any application at home—an idea captured with this oft-heard assertion: "My organization is different."
- The myth that encouraging critical thinking among the rank and file will undermine ... discipline.
- The belief that thinking inhibits doing, an idea expressed in the first century BC by the Greek philosopher Publilius Syrus. "While we stop to think," he said, "we often miss our opportunity."
- The mistaken assumption that be-

cause we have an R&D unit, we've got it covered.

- A profession that denigrates thinking about an organization's basic business establishes a culture that's likely to ridicule or demean workers who'd take time away from what the community deems "real" work.
- Reluctance to have cherished views challenged.
- Difficulty in engaging in organizational self-criticism when we know we must continue to work with the people whose current efforts are criticized.
- Insufficient time for employees to reflect on their work and a lack of time, authority, resources, and skills for them to conduct research.
- Finally, the ancient and still mighty obstacle to innovation: People usually fear change.

Few of us would argue these points. I've found these attitudes present in countless software development organizations I've visited. However, I found this list so striking because of its source. This list of impediments to fostering a learning organization is about police departments and is from an article by William A. Geller entitled "Suppose We Were Really Serious About Police Departments Becoming 'Learning Organizations'?" published in the December 1997 issue of the *National Institute of Justice Journal* (www.ojp.usdoj.gov/nij/journals/jr000234.htm).

My conclusion from this is that software development shares many of the same issues that any professional organization encounters. Therefore, we need to pay attention to the solutions suggested in other domains.

Feedback

We don't have the space here to discuss all of Geller's suggestions, but he

offers one nugget of wisdom that is so profound yet so obvious that I simply have to write about it.

Geller relates a story about an organization that initially averaged only a single suggestion per year from each employee. However, within two years, the organization succeeded in averaging one suggestion per worker per week. This remarkable improvement was achieved by instituting two policies:

- Every suggestion would get a reply within 24 hours.
- If the suggestion was good, the process for implementing it would begin within 72 hours.

Adding feedback to the first part of the learning-distribution-change chain can do wonders in getting people involved in fostering a learning organization. It helps get people to note successes and failures and spend the time to develop their observations into suggestions.

If immediate feedback can foster contributions of ideas and observations, we might speculate as to the benefits of adding feedback into the last part of the learning-distribution-change chain as well. After all, we should value attempts to apply suggestions for improvement as much as we value the original suggestions themselves—if not more.

What do you think? I'd like to hear your thoughts on the issue of becoming a learning organization. Does your organization have a planned mechanism to solicit knowledge from employees? How successful is the mechanism? What are the keys to its success? Please write to me at warren.harrison@computer.org.

EDITOR IN CHIEF:
Warren Harrison

10662 Los Vaqueros Circle
Los Alamitos, CA 90720-1314
warren.harrison@computer.org

EDITOR IN CHIEF EMERITUS:
Steve McConnell, Construx Software
stevemcc@construx.com

ASSOCIATE EDITORS IN CHIEF

Education and Training: Don Bagert, Rose-Hulman Inst. of Technology; don.bagert@rose-hulman.edu

Design: Philippe Kruchten, University of British Columbia; kruchten@ieee.org

Requirements: Christof Ebert, Alcatel; christof.ebert@alcatel.com

Management: Ann Miller, University of Missouri, Rolla; millera@ece.umar.edu

Quality: Stan Rifkin, Master Systems; sr@master-systems.com

Experience Reports: Wolfgang Strigel, Software Productivity Center; strigel@spc.ca

EDITORIAL BOARD

Nancy Eickelmann, Motorola Labs
Richard Fairley, Oregon Graduate Institute
Martin Fowler, ThoughtWorks
Jane Hayes, University of Kentucky
Andy Hunt, Pragmatic Programmers
Warren Keuffel, independent consultant
Karen Mackey, Cisco Systems
Deependra Moitra, Infosys Technologies, India
Don Reifer, Reifer Consultants
Suzanne Robertson, Atlantic Systems Guild
Richard H. Thayer, Calif. State Univ. Sacramento
Dave Thomas, Pragmatic Programmers

INDUSTRY ADVISORY BOARD

Robert Cochran, Catalyst Software (chair)
Dave Aucsmith, Microsoft
Maarten Boasson, Quaerendo Invenietis
Annie Kuntzmann-Combelles, Q-Labs
David Dorenbos, Motorola Labs
Enrique Draier, MAPA LatinAmerica
David Hsiao, Cisco Systems
Takaya Ishida, Mitsubishi Electric Corp.
Dehua Ju, ASTI Shanghai
Donna Kaspersen, Science Applications International
Pavle Knaflic, Hermes SoftLab
Wojtek Kozaczynski, Microsoft
Tomoo Matsubara, Matsubara Consulting
Masao Matsumoto, Univ. of Tsukuba
Dorothy McKinney, Lockheed Martin Space Systems
Stephen Mellor, Project Technology
Susan Mickel, Lockheed Martin
Dave Moore, Vulcan Northwest
Melissa Murphy, Sandia National Laboratories
Grant Rule, Software Measurement Services
Girish Seshagiri, Advanced Information Services
Chandra Shekaran, Microsoft
Martyn Thomas, Praxis
Rob Thomsett, The Thomsett Company
John Vu, The Boeing Company
Simon Wright, Integrated Chipware
Tsuneo Yamaura, Hitachi Software Engineering
Jeffrey Voas, Cigital

MAGAZINE OPERATIONS COMMITTEE

Bill Schilit (chair), Jean Bacon, Pradip Bose, Doris L. Carver, George Cybenko, John C. Dill, Frank E. Ferrante, Robert E. Filman, Forouzan Golshani, David Alan Grier, Rajesh Gupta, Warren Harrison, Mahadev Satyanarayanan, Nigel Shadbolt, Francis Sullivan

PUBLICATIONS BOARD

Michael R. Williams (chair), Jean Bacon, Laxmi Bhuyan, Mike Blaha, Mark Christensen, Thomas Keefe, Deependra Moitra, Sorel Reisman, John Rokne, Linda Shafer, Steven L. Tanimoto, Anand Tripathi

IN OUR NEXT ISSUE

Return on Investment in the Software Industry