Building Feedback and Natural Reinforcement

into Software Applications

By Tom Spencer, Ph.D.

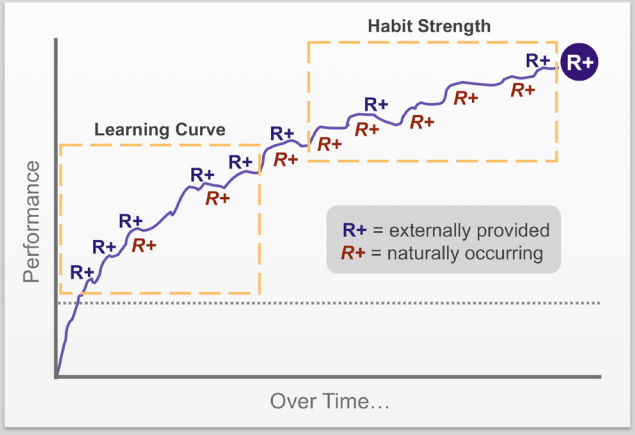
The launch of a new software application is more often met with muttering and sighs than with enthusiasm. If the new application is replacing an old one, users must not only learn the new application, they must unlearn old habits tied to the application being replaced. The un-learning occurs through extinction and typically has emotional side effects. Anyone who skipped the 2007 release of Microsoft

Building frequent PICs (positive, immediate, and certain consequences) into a software application can transform routine work into something more enjoyable or even fun.

– Tom Spencer

Office and made the leap from Office 2003 to Office 2010 experienced this directly. More recently, the launch of Windows 8 has had users clamoring for the Start menu. In both cases, fluent habits that had worked for years abruptly began leading to dead ends and frustration.

Some organizations passively manage the transition by merely removing the opportunity to use the old application, launching the new one, and then hoping for the best as users cope with the change. Many though at least provide some basic awareness training for the new application. Although rarely done, organizations should actively manage the behavior change required by providing feedback and reinforcement for using and building expertise with the new software and sharing best practices. This external feedback and positive reinforcement during skill acquisition is critical for decreasing the time to performance fluency, building software-use habits last, and appropriately acknowledging the extra time and effort caused by the software change. This approach will also prompt positive conversations about using the software while limiting the potential for negative gossip.



This figure shows how more externally provided positive reinforcement (R+) is needed during the initial stages of learning until the naturals reinforcers associated with being successful begin to maintain the performance.1

Many organizations rely heavily on "out-of-the-box" software applications to manage their core business functions, even if those applications have been somewhat customized for their business or integrated with other systems. However, out of necessity or the desire for a competitive advantage, some organizations design their own software systems or hire an external firm to do it for them. Those situations provide a unique opportunity to ensure that feedback and reinforcement are built into the application itself.

Building frequent PICs (positive, immediate, and certain consequences) into a software application can transform routine work into something more enjoyable or even fun. Video game designers, the virtuosos of frequent feedback and positive reinforcement, have demonstrated for years that people will demonstrate extraordinary Discretionary Effort™ doing what on the surface might seem like a mindless activity.

The following sets of questions were developed to help you think through how and where you would most benefit from building feedback and positive reinforcement into your software applications.

WHAT'S THE GOAL OF ADDING FEEDBACK AND REINFORCEMENT?

While it might be readily accepted that including feedback and reinforcement in the design of a software application would be a good thing to do, identifying its desired impact is critical for pinpointing where, when, and how it will be added. Potential goals for the feedback and reinforcement are listed below:

- **Improve accuracy** – What are the greatest opportunities for user error?
- **Mitigate response cost** – What actions or tasks will be the most difficult for users? The difficulty could be due to the level

of effort required, the technical complexity of the task, or the precision required. Feedback and reinforcement can help mitigate the negative effects of high response cost.

- **Improve compliance** – What actions or tasks are critical for customer service, regulatory compliance, or internal process compliance and are not being consistently performed?
- **Increase pace** – What tasks will be repeated many times in succession?
- **Improve engagement and job satisfaction** – What would the users enjoy knowing about their performance and what they are accomplishing?

Feedback and consequences intended to reinforce user behavior will quickly become annoying if they have no specific and valuable purpose. For example, if the task is easy but monotonous, users would benefit much more from feedback on their rate of task completion than on the accuracy of how they complete steps in that task. Carefully thinking through why the user will need timely feedback and reinforcement while using the application will help you earn a return on your software investment.

WHAT WILL TRIGGER THE FEEDBACK AND REINFORCEMENT?

Typically, you will arrange for the application to provide feedback and reinforcement based on an action or series of actions performed by the user, ranging from a single action to the completion of multiple tasks. You also might provide feedback and reinforcement based on other features of how they are using the application such as the validity of the data entered or the volume of tasks completed. What you will arrange to trigger the feedback or reinforcement will largely depend on your goal.

Could be triggered based on the following levels of events:

- **Individual actions** – entering text in a field, selecting a response option, etc. (e.g., entering a service code or identifying a hospital as in or out of network)
- **Set of actions** – fulfilling the requirements of a window or screen or a set of windows/screens (e.g., updating physical medicine visit calendar or completing DME window)
- **Task** – a single task such as processing a claim
- **Multiple tasks** – a series of tasks in a process or a repetition of the task with a series of inputs (e.g., processing multiple claims)

Could be triggered based on meeting one or more of the following criteria:

- **Accuracy** – the validity of data entered
- **Compliance** – compliance to a rule or set of rules (e.g., all required fields completed and response options selected based on situation, task completed within defined time period, notes screen accessed prior to performing some function)
- **Speed** – the speed of task completion
- **Productivity** – the volume of task completion

HOW CAN FEEDBACK AND REINFORCEMENT BE BUILT INTO SOFTWARE APPLICATIONS?

When building feedback and natural or engineered reinforcement into applications, ensure that the content, format, and timing and delivery method of the feedback and reinforcement are meaningful and desired by the users. Avoid frequent, modal displays that unnecessarily increase user response cost (i.e., pop-up windows that require a response to acknowledge and close).

DECREASE NEGATIVE CONSEQUENCES

- **Reduce response cost (time and effort of user)** – make the interface and functionality intuitive through simplicity, consistency, functional use of visual cues to prompt actions, and logical organization of fields, data displays, and functionality; minimize multiple requests for the same information; minimize wait time between functions; minimize use of modal dialogue boxes and alerts; include efficient search functionality; etc.
- **Reduce opportunity for error** – provide easy access to current information when it's needed to perform an action, allow actions to be undone or changed, etc.

INCREASE FEEDBACK AND POSITIVE CONSEQUENCES

- **Acknowledge actions** – use visual cues to acknowledge user actions (e.g., changes in text appearance)
- **Provide information** – provide relevant information as soon as it is meaningful and useful

HOW WILL THE INFORMATION BE VIEWED?

Consider what will trigger the feedback and reinforcement, how much information will be presented, and how the users would want to view the information you are presenting. Based on this understanding of the user experience, consider the options below and determine how you will present the information for easy and meaningful viewing by the users.

- **Non-modal window** – a pop-up window that doesn't require a user response
- **Modal window** – a pop-up window that requires a user response before the user can do anything else in the application (use with caution to avoid unintended negative consequences such as increasing response cost)
- **Current window** – a change (visual cue or text) to the current window
- **Manually accessed window** – the addition of or change to a separate, manually accessed window

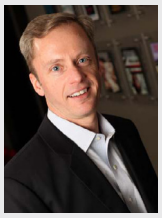
CONCLUSION

When developing or customizing software applications, think through how employees will be using the software and build the design around that expected use. Software that is easy to use (logical interface and task flow with limited response cost) and that provides timely and relevant feedback and reinforcement for the behavior you want will be quickly and painlessly adopted.

• • • • •

[About the Author]

TOM SPENCER, PH.D.



As President and CEO of Aubrey Daniels International (ADI), Tom actively works with ADI staff and clients to create positive change and achieve desired business goals. For nearly 25 years, his experience and ideas have shaped pragmatic and integrated approaches to applying the science of behavior to the workplace. Tom has written extensively on topics related to leadership, consequence management, performance fluency, and technology development. When not leading ADI, Tom enjoys trail running and following the WVU Mountaineers.

[About ADI]

Regardless of your industry or expertise, one thing remains constant: People power your business. Since 1978 Aubrey Daniels International (ADI) has been dedicated to accelerating the business and safety performance of companies worldwide by using positive, practical approaches grounded in the science of behavior and engineered to ensure long-term sustainability. ADI provides clients with the tools and methodologies to help move people toward positive, results-driven accomplishments. Our clients accelerate strategy execution while fostering employee engagement and positive accountability at all levels of their organization.

CONNECT WITH US

aubreydaniels.com/stay-connected
web: aubreydaniels.com
blog: aubreydanielsblog.com
twitter: twitter.com/aubreydaniels