Constructive Behaviorism: A Hybrid Theory and Method for Practical Use in Technical Training and Content Development for Software Companies

Barry Langer

Purdue University

Introduction

The evolution of technology is constant and ongoing. Nowhere is this more prevalent than in software. With the advent and large-scale rollout of millions of touchscreen devices, the user interface that is interacted with most consists only of virtual buttons. This is accompanied by a proliferation of devices of multiple sizes, ranging from pocket-sized to 80", or even to an all-encompassing virtual reality experience.

From a developer's perspective, much of the weightier on-premises hardware obligations have been replaced by cloud-centered technologies. This is a shift of equal weight, as cloud-based applications allow users to avoid the "black box" of engineering the infrastructure for that app, instead giving users the opportunity to concentrate more fully on the tasks facing them instead of physical setup and maintenance.

The complexity of the tasks facing the users presents a fundamental usability issue. How can a software development company create a piece of software that is easy to use, easier to learn, and still retains the same kinds of capabilities as less friendly competitors? A designer will not want to sacrifice the beauty of their modern interface, while an engineer will not want to remove any features that could provide the user with added value. Finding the balance is difficult, especially during the continued adoption of cloud back-ends.

The trend towards simplifying user interfaces and deployment techniques creates an opportunity for software trainers. We must find ways to provide our users with the correct behaviors they will need so that they can be successful. We do not want to simply have them walk through a simple series of rote steps every time; if we do this, we will find groups of users marooned without while allowing them the right proportion of freedom to discover how to do these tasks themselves. This paper aims to specifically approach this issue using selected pieces

of constructivism and behaviorism. These two contrasting theories can be blended together to create positive learning experiences for the relevant parts of a product lifecycle – that is, onboarding and daily use. This hybrid theory of constructive behaviorism, where a sliding scale of behaviorist techniques and constructivist ones blend together into an amalgam, is applicable to all types of software training, though is intended for a target audience of adults rather than children.

Two Distinct Theories

It would not appear that constructivism and behaviorism would make for an obvious pairing. Behaviorism uses a well-defined standard to determine whether "a proper response is demonstrated following the presentation of a specific environmental stimulus" (Ertmer & Newby, 2013, p. 48). Behaviorism treats the learner as the recipient of the information, and has them engaging in a more passive manner with the material; the only way for the learner to demonstrate anything they have learned is through recreation of the tasks set out before them (Driscoll, 2005, p. 58-59). This creates a learning environment where the content being acquired creates certain "types of association" for "learning about the relationship between events in [a learner's] environment" (Graham, 2017). This allows for learners to be self-sufficient when presented with goals, and to follow a series of instructions towards an obvious end.

By contrast, constructivism advocates for learner-defined meaning to be wrung from the presented materials, that "values the students' points of view and attempts to encourage students in the directions they have charted for themselves" (Brooks & Brooks, 1999, p. 5). The value that behaviorism places on the successful reproduction of behaviors does not apply. Instead, learners create significance in the context of the learning environment, with the ideas of "ownership" and "self-awareness" taking on positions of high priority (Driscoll, 2005, p. 390-

391). This works well within a social, collaborative learning environment, especially when it can involve individuals working together to accomplish general growth.

Connecting Our Theories to Software Training

The differences between the two theories are rather stark. While behaviorism pushes for learning "in response to stimuli," (Graham, 2017) and the creation of meaning in pieces through the repetition of processes, constructivism pushes against the use of a required order of steps. This preferences towards a learner creating their own meaning, and therefore deriving a more complete understanding of the whole, rather than just the individual parts, allows for greater building of conceptual knowledge (Brooks & Brooks, 1999, p. 46-47). Both halves are of the highest importance when learning any new piece of software. To understand how an individual task is done is acceptable, and to be able to intuit how certain expected actions will work is crucial. Combining these two philosophies creates a fusion where the individual is taught how to intuit certain larger concepts through the understanding gained from specific behaviors.

The ways in which these behaviors are taught is also of vital importance. If a software trainer were to simply release a group of professionals onto a new product, the intent would be to allow them to guide themselves to a final goal of understanding that product. However, as Kirschner, Sweller, & Clark (2006) state,

> Most learners of all ages know how to construct knowledge when given adequate information and there is no evidence that presenting them with partial information enhances their ability to construct a representation more than giving them full information…Learners must construct a mental representation or schema irrespective of whether they are given complete or partial information. (p. 78)

There is little that is worse for a paid training session – especially one with a goal of certification on a software title in mind – than to have the user come out feeling they have been given an incomplete experience. This would cause them to seek out new sources of information, with the potential of leaving negative feedback. This is especially true during an initial user onboarding, when their experience with the product is still in its infancy.

Onboarding a new user to a product will also require that the curriculum provide ample motivational pushes for the user to acquire new skills. The use of more heavy-handed instruction that emphasizes behaviors based on constructed stimuli (e.g., tapping on the menu in the top left corner brings up a sidebar of options on the left-hand side of the screen) needs a user to maintain sustained attention throughout the duration of the training. This is important across all learning experiences (Keller, 1987, p. 3), but at this stage of training, getting a user invested in the product will determine the rest of their experience using it.

The current user of the product is likely to be the most important part of the trainer's job. Trainers are not usually involved in the customer acquisition process, even if the promise of training is part of what makes a title compelling to a new customer. The training materials that are produced are likely to be used at least partially towards the procurement of a new customer. Marketing materials often are too soft in their messaging for members of technical staff, and technical training videos, walkthroughs, or self-guided hands-on labs may be accessed by potential clients. These uses of software training materials merit consideration, even if they are not the primary goal of the training. The idea behind supplying a prospect with these kinds of materials is to create a positive feeling within the target users of that product within the potential customer. Using constructivist learning media places the onus of contextualization on those users (Driscoll, 2005, p. 398-399), and lets them determine which ways the product can benefit

them in future uses. This creates a situation where they feel an "ownership in learning" (Driscoll, 2005, p. 399) that pushes them past interest and into advocacy.

Gaining a user advocate for a software product should be a priority of any software trainer. By leaning more heavily on behavioral cues at the beginning – specifically those designed to create larger meaning around a cloud application's modern user interface – the user will come away with a deliberate skillset that self-guided learning cannot accomplish on its own. It is at this stage that we can simply theorize about undesirable task sequences, and hope to quash any undesirable performances (Mager & Pipe, 1997, p. 61-62) at the outset.

This is not always going to be tenable; consider the needs of a seasoned user returning for recertification. The ratio of instruction will necessarily tip more heavily towards constructivist methods to reinforce the behaviors already understood by that user. This is where we will see a greater use of relevance strategies to reinforce the "present intrinsic value of learning the content, as distinct from its value as a link to future goals. (Keller, 1987, p. 4)." Trainers will need to investigate if any of the users have already found workarounds that they are leveraging that lead to favorable results. If so, finding ways to remove these behaviors through detailed explanation as to why they are unsustainable (Mager & Pipe, 1997, p. 43-44). This is more easily accomplished with small group activities, where the learning activities are formulated in such a way that the easiest solution is also the best practice.

Using Constructive Behaviorism in a Lone User Scenario

As this is a new theory, there are no established guidelines for how any one solution would work. Constructivism can work well to establish rapport among learners with technological tools (Driscoll, 2005, p. 406), while behaviorism would rather users shape understanding through a series of actions to learn that same product (Driscoll, 2005, p. 44). As is

common with technology, there have been advances in software learning experiences that do fall directly in line with constructive behaviorism. One such example, You Need A Budget (YNAB, 2017) is structured as a guided tour of the product that demands the user take action. After signing up for a free trial, the user is provided with a set of screen overlays that guide them through various features that support the use of this personal financial management software. These overlays allow the user to interact within the software, while supporting interface text provides context to the user as to what is happening. The overlays do not simply point out the features of the budget screen or where transactions would be entered, but rather prompt the user towards the behavior of adding their current bank accounts, setting up organizational rules, and creating categories that they can use to build their new budget (Tran, 2015). The app allows the user to proceed at their own pace, and they can jump through the steps in the workflow to gain greater understanding. If the user follows the prescribed steps, then they will take the action and complete the intended behavior. It is the combination of self-guidance with clear behavioral expectations that makes this experience so powerful.

The first user experience with any product is considered so meaningful that multiple tools to guide users through the initial use have been developed. Chameleon, Chaperone, Userlane, Nickelled, Appcues, Inline Manual, AppCues and onboardX specialize in assistive tours through webapps, all with the same goal of easing users into a product with the hopes of retaining them past a trial period (AlternativeTo, 2017). These methods lean more heavily on the behavioral, but when implemented using constructivist and behaviorist techniques, users can encode more meaningful memories due to the significance they attached to building their skillset. An increase in the use of these types of guided tutorials is a key way to train users of any software without having to involve a live trainer and could be a first step towards success for users of any product.

Conclusion

Constructivism on its own provides an excellent groundwork for building memorable learning experiences among a group of individuals. It does not provide these learners with enough concrete guidance to work in a world of software that is rapidly advancing. Behaviorism can explain the right actions that a user of a new or updated piece of software needs to take, but simply walking a user through a stock series of actions does not create an experience that the user will retain without heavy repetition. By combining the principles of constructivism with those of behaviorism, we can find a happy medium for users of any software title. Modern tools enable this to quickly jumpstart in meaningful ways and can assist users in naturally recalling their newly acquired skills.

References

Brooks, J.G., & Brooks, M.G. (1999). *In search of understanding: the case for constructivist classrooms*. Alexandria, VA: Association for Supervision and Curriculum Development.

AlternativeTo. (2017). Retrieved February 28, 2017 from https://alternativeto.net/software/chameleon/

Driscoll, M.P. (2005). *Psychology of learning for instruction* (3rd ed.). Boston, MA: Pearson, Allyn, and Bacon.

Ertmer, P.A., & Newby, T.J. (2013). Behaviorism, Cognitivism, Constructivism: Comparing Critical Features from an Instructional Design Perspective. *Performance Improvement Quarterly, 26*(2), 43-71.

Graham, G. (2005, March 11). Behaviorism. In E.N. Zalta (Ed.), *Stanford Encyclopedia of Philosophy*. Retrieved February 26, 2017 from forthcoming URL https://plato.stanford.edu/archives/spr2017/entries/behaviorism/

Keller, J.M. (1987). Development and use of the ARCS model of instructional design. *Journal of Instructional Development, 10*(3), 2-10.

Kirschner, P.A., Sweller, J., & Clark, R.E. (2006). Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching. *Educational Psychologist, 41*(2), 75-86.

Mager, R.F., & Pipe, P. (1997). *Analyzing performance problems or you really oughta wanna* (3rd ed.). Atlanta, GA: The Center for Effective Performance, LLC.

Tran, A. (2015). *You Need A Budget Walkthrough.* Retrieved February 27, 2017 from https://www.alextran.org/wp-content/uploads/2015/06/ynab-walkthrough.gif

You Need A Budget. (2017). Retrieved February 27, 2017 from

https://www.youneedabudget.com