When Updates Make a User Stick: Software Feature Updates and their Differential Effects on Users’ Continuance Intentions

Completed Research Paper

Marvin Fleischmann
Ludwig-Maximilians-Universität München
Ludwigstr. 28, 80539 Munich, Germany
fleischmann@bwl.lmu.de

Tillmann Grupp
Technische Universität Darmstadt
Hochschulstraße 1, 64289 Darmstadt
grupp@ise.tu-darmstadt.de

Miglena Amirpur
Technische Universität Darmstadt
Hochschulstraße 1, 64289 Darmstadt
amirpur@ise.tu-darmstadt.de

Alexander Benlian
Technische Universität Darmstadt
Hochschulstraße 1, 64289 Darmstadt
benlian@ise.tu-darmstadt.de

Thomas Hess
Ludwig-Maximilians-Universität München
Ludwigstr. 28, 80539 Munich, Germany
thess@bwl.lmu.de

Abstract

Although software updates are extensively used to enhance software while already being used, their impact on users’ post-adoption beliefs and attitudes has received little attention. Drawing on expectation-confirmation-theory and the IS continuance model, we investigate if and how feature updates affect users’ continuance intentions (CI) and what role initial feature endowment and update size play. In an online experiment, we find a positive effect of feature updates on users’ CI. According to this effect, software vendors can increase users’ CI by delivering features later, through updates instead of providing them right with the first release. While this positive effect persists despite a small update size and high initial feature endowment, the latter diminishes the effect. We also unveil positive disconfirmation of previous expectations regarding the updated software as crucial mediating mechanism between feature updates and CI. Implications for research and practice as well as directions for future research are discussed.

Keywords: Feature updates, continuance intentions, feature endowment, update size, expectation-confirmation-theory, IS post-adoption theory
Introduction

In many cases, software vendors nowadays no longer sell their applications as monolithic packages but instead constantly enhance and extend their products after their first release and while the software is already in use by their customers. This is a phenomenon that is particularly prevalent in the field of consumer software such as apps for smartphones and tablet computers but also applies to desktop computer software and web services. For example, Microsoft’s Office 365-Suite received 127 updates since its release in June 2011 (Microsoft 2015). Another example is Facebook. The popular social network received over ten major software feature enhancements (e.g., keyword search in all posts and read-it-later-feature) only in 2014 (Facebook 2015). In the case of such ‘agile software’ (Hong et al. 2011), vendors have to make two strategic decisions regarding the implementation of software features. First, they have to decide what and how many functionalities the first release of the software should comprise, i.e., at what stage of development the software should be released. Second, after this first release, vendors have to decide how to deliver the new features that result from the ongoing development. Functionality that is delivered after the first release is usually delivered to users through free ‘feature updates’. These feature updates are no discrete and standalone programs themselves but are rather integrated into the base software once they are applied to it (e.g., Dunn 2004). In practice, features are sometimes delivered in individual updates or in larger update-packages, containing several new features at once.

For vendors of agile software, it is important to understand how their customers perceive these feature updates. Because they alter software during use, feature updates may impact users’ post-adoption beliefs and attitudes regarding the software and thus even affect their intentions to continue using the software (Hong et al. 2011). More specifically, the continued use of software by users (i.e., customer loyalty) has become an important goal for vendors because business models in the software industry increasingly rely on recurring revenues from subscriptions or the sale of ads and therefore a large and active user base. However, despite this common practice of extending and enhancing software during use and its potential impact on vendors’ revenues, the effect of feature updates on users’ continuance intentions (CI) remains largely unexplored. While there is an extensive body of research on software engineering (Sommerville 2010), software product lines (Clements and Northrop 2002), software release planning (Svahnberg et al. 2010) and software evolution and maintenance (Mens and Demeyer 2008), this primarily addresses technical considerations. Post-adoption research which explores the user’s perspective, on the other hand, often tends to conceptualize information systems as a monolithic and coarse-grained black box, rather than as modular composition of specific and finer-grained features which may be altered after the software’s first release (e.g., Bhattacherjee 2001). Even the few post-adoption studies that do consider changes in the software’s functionality after its first release do not account for different ways of delivering features (e.g., the number of features delivered in one update) and a potential interaction with the initial feature endowment (i.e., a software’s level of functionality at its first release) (Hong et al. 2011). To address these shortcomings, we will study the impact of different feature delivery strategies on users’ continuance intentions regarding software in non-mandatory usage settings (e.g., software use by consumers). Specifically, we address the following research questions:

RQ1: How and why do feature updates impact users’ continuance intentions regarding software?

RQ2: How do initial feature endowment and update size affect the potential impact of feature updates on users’ continuance intentions?

To answer these questions, we conducted a vignette-based online experiment with 261 participants, allowing us to identify and isolate causal evidence of the effect of feature updates on users’ CI. In doing so, our study contributes to post-adoption research in three considerable ways. First, we identify a somewhat counter-intuitive, positive effect on user’s CI from a deferred delivery of software features. Users who receive features through updates during software use have higher CI than users who have access to all features right from the beginning of their software usage. This phenomenon seems to be robust to manipulations of update size (i.e., the number of features delivered in one update) and occurs even under a high initial feature endowment. However, the positive effect seems to diminish with increasing initial feature endowment. Second, by disclosing the relative nature of positive disconfirmation of (subjective) previous expectations regarding the software as the mediating mechanism behind this positive effect of updates, we find a possible empirical evidence for reference point dependency in users’ perception of
software (Kahneman and Tversky 1979). Third, we advance the understanding of IS post-adoption behavior by conceptualizing and exploring information systems as a fine-grained, malleable and dynamic collection of modular features rather than a monolithic block which is static over time. From a practitioner's perspective, our study offers important implications for vendors of consumer software. First, we describe how vendors might increase their customers’ loyalty by strategically deferring the delivery of features through the use of software updates. Moreover, if holding back features is strategically not feasible (e.g., due to competition) our findings also suggest that it still beneficial to release a software early on and only subsequently roll out additional features, once they are developed. Our study also suggests that this measure should work with software that has a low initial feature endowment as well as for more mature and feature-rich software. Ruling out update size as potential boundary condition to this effect moreover implies that vendors should deliver feature innovations individually, instead of bundling them in large update packages.

**Theoretical Foundations**

**Feature Updates**

Consistent with previous research (e.g., Dunn 2004), software updates can be defined as self-contained modules of software that are provided to the user for free in order to modify or extend software after it has been rolled out and is already in use. Software updates are no discrete and stand-alone programs themselves, but rather integrate into the software to which they are applied. With varying terminology (e.g., update, upgrade, patch, bug fix, or hotfix), the concept of software updates is repeatedly addressed throughout the software engineering literature (Sommerville 2010). This includes software release planning, software maintenance and evolution and software product lines (Shirabad et al. 2001; Svahnberg et al. 2010; Weyns et al. 2011). In this context, software release planning or strategic release planning refers to the “idea of selecting the optimum set of features or requirements to deliver in a release within given constraints” (Svahnberg et al. 2010, p. 1). Following this definition, an update is the delivery of features after the first release of a software and falls within a vendor’s strategic considerations regarding when to deliver what type of functionality to the user. Literature on software evolution and maintenance addresses the later stages in the software lifecycle, where updates are utilized to adjust software to changing requirements or repair emerging flaws in the software while it is already in use (Shirabad et al. 2001). In contrast to this rich stream of technical literature dealing with software updates, research on user’s beliefs and attitudes regarding updates has so far been very limited (Amirpur et al. 2015). Hong et al. (2011), for example, explore users’ acceptance of information systems that frequently change through the addition of new functionality. And while Benlian (2015) examines IT feature repertoires and their impact on individual task performance, he does not consider changes in these repertoires through updates. Other studies that investigate updates have often pushed them to the sidelines, treating them as control variables for studying other phenomena (e.g., Claussen et al. 2013). Existing IS research has, however, not explored the specific impact of updates on users’ perceptions of an IS. Specifically, essential system characteristics such as the pre-update feature endowment of a software or the number of features in one update have so far not been explored in this context.

For the purpose of this study, we distinguish two basic types of software updates: feature updates and non-feature updates. Feature updates change the core functionality of the software to which they are applied. Functionality thereby refers to distinct, discernible features which are deliberately employed by the user in accomplishing the task or goal for which he or she uses the software. In contrast to feature updates, technical non-feature updates do not change the core functionality of software but only correct flaws or change software properties. Non-feature updates usually do not directly affect the user’s interaction with the software and are typically not even visible to the user (e.g., improvements in stability, security or performance) (Popović et al. 2001). Because the core functionality is frequently utilized for accomplishing the task for which the software is used, a change in the software induced by feature updates is most often notable for users. If the software’s core functionality is changed, the user’s interaction with the software may also change. As we will outline later on, we argue that feature updates thus have the potential to influence users’ beliefs, attitudes, and behaviors regarding the updated software in the post-adoption stage of IS usage. This may even affect their decisions on continued use or discontinuation. Before further substantiating this claim, we proceed by reviewing research on IS continuance.
Information Systems Continuance

Post-adoptive research studies users’ beliefs, attitudes, and behaviors after the initial adoption of an IS (Benlian et al. 2011; Bhattacharjee 2001; Karahanna et al. 1999). One of the main goals of post-adoptive research is the exploration of users’ information system continuance, which is defined as the “sustained use of an IT by individual users over the long-term after their initial acceptance” (Bhattacharjee and Barfar 2011, p. 2). To explore users’ continuance behaviors, Bhattacharjee (2001) adopts expectation-confirmation theory (ECT) (Anderson and Sullivan 1993; Locke 1976; Oliver 1980; Oliver 1993) and proposes a model to explain users’ intentions to continue using an information system as a result of satisfaction (SAT), perceived usefulness (PU) which are in turn determined by a confirmation or disconfirmation of previous expectations regarding the software (DISC). Following ECT, the IS continuance model suggests that users compare their pre-usage expectations of an IS with their perception of the performance of this IS during actual usage (Bhattacharjee 2001). This comparison of expectations with usage experiences has also been shown to occur in later stages of use, where expectations are sequentially updated through ongoing usage experiences Kim and Malhotra (2005). If perceived performance exceeds expectations, users experience positive disconfirmation (DISC) which has a positive impact on their satisfaction with the IS. If, on the other hand, perceived performance falls short of the expectations, negative disconfirmation occurs and users are dissatisfied with the IS (Bhattacharjee and Barfar 2011). Positive (negative) disconfirmation thus comprises two essential elements: unexpectedness and a positive (negative) experience. ECT moreover posits expectations as a relative reference point or baseline (i.e., not an absolute, objective one) upon which the user makes a comparative judgment (Oliver 1980). This idea of a subjective, relative reference point is based on Helson’s (1964) adaptation level theory, which proposes that human beings perceive stimuli relative to or as a deviation from an ‘adapted level’ or baseline stimulus level. “This adapted level is determined by the nature of the stimulus, the psychological characteristics of the individual experiencing that stimulus, and situational context” (Bhattacharjee 2001, p. 354).

While applications of the continuance model have made valuable contributions in exploring post-adoptive phenomena, IS researchers often tend to conceptualize the studied information systems as a monolithic and coarse-grained black box, rather than as collection of specific and finer-grained features that are alterable after the first release. However, accounting for the granularity of software would help to explain how users respond to different compositions of software features and how changes in this composition through e.g., software updates after the first release might affect users’ beliefs, attitudes, and behaviors regarding an information system (Benlian 2015). In addition, there are several calls for research from IS scholars who criticize the negligence of the IT artifact’s role in IS research (Benbasat and Zmud 2003; Hevner et al. 2004; Orlikowski and Iacono 2001). They advocate for focusing more on changes in users’ beliefs, attitudes and behaviors, emanating from the IT artifact itself rather than from other IT-unrelated environmental stimuli. By studying the impact of software updates on users’ continuance intentions, we account for this malleable nature of the IT artifact and address these calls for research.

Hypothesis Development

Our study is motivated by the overarching idea that the interplay between initial feature endowment and post-release update size may impact users’ continuance intentions regarding an IS. The expectation-confirmation mechanism (Oliver 1980) incorporated in the IS continuance model (Bhattacharjee 2001) serves as theoretical lens through which we will investigate the roles of feature endowment and update size and develop our hypotheses. When receiving updates during use of an IS, ECT implies that users’ continuance intentions crucially depend on a comparison between pre-update expectations and post-update experiences with the IS. Specifically, we theorize (1) how a deferred delivery of features through updates can increase users’ continuance intentions, how this might be affected by (2) initial (pre-update) feature endowment and (3) the number of features delivered in a post-release update (i.e., update size). We also hypothesize how this proposed effect is mediated through a chain of relations, initiated by a positive disconfirmation of previous expectations (4).

As highlighted in the introduction, when considering software as a flexible combination of modular functionalities, a vendor of an agile information system has to balance two crucial design parameters and their reciprocal interaction: (1) the initial set of features (hereafter called feature endowment) of a
software at the time of its first release and (2) the set of features which is added to the software through updates after its first release (hereafter called update size). We investigate how a given set of features can be balanced between the initial release of a software and the later delivery through updates and how this design choice affects users’ continuance intentions regarding the software. The settings which we investigate are widely used in practice (and promise interesting theoretical insights). They comprise (1) a low initial feature endowment and large update size, (2) a high initial feature endowment and a small update size and (3) a low initial feature endowment and a small update size.

Our theorizing about the differential allocation of features between the initial release and the later delivery requires the assumption that each feature provides an equal value to the user. This is necessary, because features with different levels of importance could interfere with our attempts to conceptually isolate the effects of different feature allocations between initial release and later update on users’ CI. While being restrictive, this is a necessary assumption for identifying the proposed causal effects. In addition, we assume that the number of planned features for a certain time period is predetermined. This time period may vary depending on the planning period of a vendor in which release decisions are taken. Moreover, because we investigate users’ perceptions during usage, we focus on feature updates with explicit user notification and neglect silent, background updates which are unnoticed by the user. Lastly, we will focus on updates for consumer software applications with an unobtrusive update process (in contrast to e.g., some desktop operating system updates processes where system usage may be severely disturbed by updates).

**The Effect of Feature Updates on Users’ Continuance Intentions**

We argue that feature updates, which provide additional functionality that directly serves users in accomplishing their IS-based tasks, will be perceived as a positive experience with the software. Furthermore, it is reasonable to assume that feature updates are usually not anticipated by users and are thus unexpected experiences with the software. According to ECT, if feature updates are perceived as unexpected and positive experiences during usage, they should consequently induce perceived positive disconfirmation (Anderson and Sullivan 1993). Following the IS continuance model (Bhattacherjee 2001), it is plausible to assume that this perceived positive disconfirmation during software use will increase users’ CI regarding the updated software. We will further elaborate on these arguments below.

In the context of software features, ECT also implies that positive disconfirmation from a feature update results from a relative change in functionality compared to the user’s subjective reference point (i.e., the pre-update configuration of the software) rather than an absolute change (Helson 1964; Oliver 1980). A software vendor should thus be able to induce positive disconfirmation and therefore increase the user’s CI by applying the strategy of simply holding back features (functionality) in the first release of the software and delivering this functionality only later on, through feature updates (Sen and Morwitz 1996). Under this deferred feature delivery strategy, a feature-complete software package can be designed and developed by the software vendor, but certain features might be removed from the initially shipped version of the software. The user is usually unaware of the existence of these remaining features. Once they are delivered through an update, they likely elicit positive disconfirmation because the user may perceive them as a ‘gift’ from the vendor. Consistent with the IS continuance model, this could then lead to an increase in CI (Bhattacherjee and Barfar 2011). This deferred feature delivery strategy is thus to be distinguished from an all-at-once feature delivery strategy under which all developed features are delivered in the first release. Nonetheless, both feature delivery strategies are assumed to comprise the same type and number of features overall.

Regarding our assumptions about feature updates, we acknowledge that in practice, there might be cases, where feature updates are perceived negatively by users. For example, if features are intentionally removed (e.g., because of expired licensing deals), software functionality is unintentionally impaired or if updates bring major changes to the software which necessitates users to learn and adjust (Polites and Karahanna 2012; Mukherjee and Hoyer 2001). Nevertheless, we argue that in most cases, feature updates are intended to enhance the software with regard to its core purpose and are thus perceived positively (Larsen et al. 2009). Furthermore, it is reasonable to assume that users perceive updates as unexpected events during usage. In many cases, updates are not announced beforehand and even if software vendors announce release plans about future updates, most end-users (and consumers in particular) likely do not follow them in detail. Moreover, if a feature is delivered through an update, it may ‘stick out’ more than if
When Updates Make a User Stick

it is included in the first release where it is one among many other features. The positive perception of this feature received through an update and its effect on CI may thus be increased even further. To summarize, because of the nature of the disconfirmation mechanism in ECT, which operates through an evaluation of relative instead of absolute change (Oliver 1980), users who receive functionality through feature updates will likely have a higher intention to continue using a software than users who received all these features right with the first release. Accordingly, we propose our first hypothesis:

**H1:** Software that receives functionality through feature updates induces a higher continuance intention compared to software that includes all this functionality right with the first release.

**The Role of Initial Feature Endowment**

As discussed before, vendors of agile software can decide what features to include in the first release and what features to deliver only later on, after the first release. Thus, in the first release, different levels of feature endowment are possible. A vendor might decide for a small initial feature set or for a more comprehensive set of features. In both cases, additional features may be added to the software through updates after its first release. Then, the possibility arises that the previously proposed positive effect of feature updates on users’ continuance intentions might only occur under a low initial feature endowment and that it might disappear under high initial feature endowment. One reason for this that the addition of one feature to an already well endowed software might be perceived as negligible by users.

As implied by ECT, the positive effect of feature updates on users’ continuance intentions requires—in addition to unexpectedness—a positive experience (Oliver 1980). In the context of software features, this experience is positive when it exceeds previous expectations regarding the functional capabilities of the information system (i.e., the subjective reference point) (Helson 1964). Users form this subjective reference point based on their overall pre-update experience with the software. We acknowledge that competing software could serve as objective reference for user’s evaluations of feature endowment. However, in practice, competing software is usually sufficiently differentiated with regard to functionality (i.e., does not have identical functionality) to prohibit a direct and objective comparison of feature endowment. Moreover, users (and consumers in particular) likely do not know about each individual feature that is available in every other software product on the market. Furthermore, feature updates usually provide functionalities which are unique and serve a distinct purpose, making them different from any other feature that is already included in the initial release of the software. Because disconfirmation is based on a relative comparison of the pre-update and post-update state of the software, any feature update that improves the functional capabilities of the software will likely induce positive disconfirmation—indepedent from the factual feature endowment of the software in its pre-update state (Anderson and Sullivan 1993). We thus suggest that even if software has a high initial feature endowment and receives a feature update, users’ CI will nonetheless increase compared to software that provides all these features right with the first release. To summarize, from the perspective of an objective evaluation one might expect a different outcome, but the outlined theory suggests that the positive effect from a deferred delivery of features through updates persists despite a high initial feature endowment. We argue that this is the result of the interplay between users’ subjective evaluations of feature endowment, the relative nature of the ECT mechanism and the functional uniqueness of features. We thus hypothesize:

**H2:** The higher continuance intention induced by feature updates is independent of the initial feature endowment of the software.

**The Role of Update Size**

From an objective point of view, one might also expect users to value larger update packages that comprise many features (i.e., large update size) more than smaller update packages which contain only a few features (i.e., small update size). However, as stated above, ECT implies that positive disconfirmation depends on a relative change in functionality compared to the user’s reference point rather than on an absolute change (Helson 1964; Oliver 1980). In the ECT mechanism, this subjective evaluation does not only comprise the pre-update state of the software which forms the baseline for the comparison (i.e., the reference point) but also the post-update state of the software and the evaluation of the feature update itself (Oliver 1980). We thus argue that while users will likely perceive a feature update as a ‘free gift’ from
the vendor, they are not likely to evaluate the magnitude of its functional value objectively. Because users do usually not expect an update and have no objective comparison regarding the comprised features, any update—irrespective of the number of contained features—will likely induce the positive effect as suggested in hypothesis 1. Moreover, we argue that an objective evaluation of an update through the comparison with similar, competing software as discussed in hypothesis 2 is even less likely to occur in practice. This is, because competing software differs even more with regard to the timeframes and extent of feature delivery through updates than it does with regard to initial feature endowment. Therefore, we argue that it is likely that users will not be able to acknowledge the distinction between large and small update sizes. We suggest that the magnitude of the effect of disconfirmation should thus not depend on update size. To sum up, it is reasonable to assume that the number of features in an update is likely to be neglected when users’ continuance intentions are formed after an update. We thus hypothesize:

\[ H3: \text{The higher continuance intention induced by feature updates is independent of the update size.} \]

**The Mediating Effect of Disconfirmation**

As outlined before, we suggest that feature updates work through a disconfirmation of previous expectations regarding the software (Oliver 1980; Anderson and Sullivan 1993). In terms of the continuance model, disconfirmation should thus mediate the effect of feature updates on continuance intentions (Bhattacherjee 2001). The IS continuance model further posits, that this disconfirmation in turn impacts perceived usefulness and satisfaction of an information system (Bhattacherjee and Barfar 2011). Both, satisfaction and perceived usefulness will furthermore drive users’ intentions to continue using an updated IS. We therefore argue that the higher levels of CI regarding software that receives functionality through feature updates compared to software that includes all features right with the first release, will likely be the result of the mediating effect of DISC and in turn affect the downstream variables of the IS continuance model (i.e., perceived usefulness and satisfaction). We thus hypothesize:

\[ H4: \text{The positive effect of feature updates on continuance intentions is mediated by disconfirmation of initial expectations regarding the software.} \]

**Method**

**Experimental Design**

To examine the effect of feature updates on users’ CI and the roles of initial feature endowment and update size as suggested by our hypotheses, we conducted a vignette-based online experiment with manipulations of update size (small vs. large) and initial feature endowment (low vs. high). Additionally, we controlled for the user’s construal level. Construal level is an individual’s mental mode of decision making and has been shown to crucially influence decisions depending on the subject matter’s perceived levels of abstraction and detail (Trope and Liberman 2003). According to construal level theory, individuals with high construal levels think more abstractly, focus on bigger picture concerns and show less myopia by pursuing long term goals compared to individuals with low construal levels. These individuals are more focused on short term interests and are subject to myopia (Fujita et al. 2006; Mehta et al. 2014; Wan and Agrawal 2011). We deliberately controlled for these different ways in thinking because we sought to show that our hypothesized negligence of initial feature endowment (H2) and update size (H3) even holds true for these two modes of thinking which seem likely to affect the perception of different allocations of features (i.e., software capabilities) between initial release and later updates. Specifically, individuals with high construal levels and long term orientations might favor software that receives features through updates over its usage cycle. A short term orientation from low construal levels, on the other hand, might lead to a preference of high initial feature endowments because this might satisfy short term goals.

To account for the different, feasible delivery strategies for a given set of features as proposed in our hypotheses, we defined one control group and three incomplete factorials. The groups had (A) all features right with the first release and no updates (control), (B) low initial feature endowment and large update size, (C) high initial feature endowment and small update size, (D) low initial feature endowment and small update size. For each of these groups, construal level (low vs. high) was additionally manipulated.
though textual vignette treatments. The used software and the task for which the software had to be used were deliberately held constant across all conditions. Overall, this lead to the 4x2 between-subjects design depicted in Figure 1. Subjects were randomly assigned to one of the eight groups. We realized the manipulations of feature endowment, update size and construal level by presenting participants with carefully constructed textual scenarios (vignettes) that precisely described a person (user), task and software characteristics (vignette setting), software usage and a conditional update during usage (vignette usage) (see Figure 1). The experimental vignette methodology (EVM) (e.g., Aguinis and Bradley 2014) was used because it provided us with the possibility to control for the user’s construal level, avoid social desirability bias and eliminate undesired learning effects of participants. Even though this method comes with some downsides such as simplifications and constructed hypothetical usage scenarios, it also enabled us to isolate and precisely manipulate the dependent variables while nevertheless accurately identifying the hypothesized causal relationships. Compared to a laboratory experiment that e.g., uses simplified prototype software as a treatment, we favored vignettes to achieve a high external validity by being able to design a realistic scenario. Researchers in IS and other disciplines have repeatedly shown that individuals respond quite similarly whether they are presented with a hypothetical situation using vignettes or a hypothetical situation in a traditional laboratory experiment. Therefore a scenario based manipulation can be assumed to work appropriately if constructed carefully (De Cremer et al. 2007; Dennis et al. 2012; Rahman 1996; Shaw et al. 2003). This makes the method suitable for our needs.

![Figure 1. Experimental Procedure](image)

The experiment proceeded in four major steps: First, upon arrival at the website, subjects were told to read instructions carefully and to answer questions to the best of their knowledge, followed by questions about subjects’ motivation to process information. Second, subjects were randomly assigned to one of the eight experimental groups and then presented with the corresponding vignettes (see Figure 1). We instructed subjects to carefully read the vignettes and put themselves in the hypothetical setting described in the scenario, before answering the subsequent questions. The vignettes described a user in a high or low construal state of mind (person), a travel booking task (task) and a travel booking platform (TBP) including its initial feature endowment depending on the experimental condition (software). Third, on the next page, subjects were presented with part two of the vignette. This part described how the person in the scenario uses the TBP to accomplish his task. Halfway through the usage time, the availability of new functionality through an update was described according to the respective experimental condition (not applicable for the control group). After the update, this part of the vignette ended with further description of TBP usage up until the task was completed. Finally, a post-experimental survey on the following pages asked subjects to respond to questions measuring their evaluation of the CI of the person described in the scenario with regard to the TBP and all further variables (see Measures). On the last page of the survey, subjects were debriefed and thanked for their participation.
Manipulation of Independent Variables

Regarding the described software, for our experiment we opted for an online travel booking platform because we wanted to ensure that subjects had previous usage experience and would thus understand the outlined scenario quickly and well. Moreover, we had the goal that the findings regarding updates which we obtained from this exemplarily setting could be generalized across a wide range of software types. The choice of an online service (i.e., the travel booking platform) allowed us to isolate the effect of receiving features through updates from other intervening factors such as waiting times or technical difficulties during the installation of updates which might occur on e.g., desktop computers (Sykes 2011; Tyre and Orlikowski 1994). These may have traditionally been issues associated with updates. However, we argue that such technical downsides of software updates have been reduced over time and have mostly disappeared in online services (e.g., Facebook), platforms for mobile devices (e.g., Apple iOS) and modern desktop operating systems (e.g., Microsoft Windows 10) where updates are now mostly unobtrusive and frictionless. We are thus confident, that with the described travel booking platform, we can derive viable implications that are generalizable across a large part of the modern software market.

The specific task was booking a two-week vacation including flight and hotel with a limited monetary budget and further constraints which fostered the use of the individual features on the TBP. The person described in the scenario was a student called ‘Tom’ and the travel booking task was chosen to be typical for a student. To construct the different stimuli with respect to update size and initial feature endowment, we identified features of a TBP that satisfied three criteria: 1) they needed to be self-explanatory, 2) they had to be perceived as useful for the task by participants, 3) when absent, the TBP still needed to be functional and the general task—while being more difficult—could still be accomplished. Through interviews and research, we compiled 22 features that meet these criteria. The importance of each feature was evaluated in a pre-study¹. Eventually, seven features were identified as appropriate and relevant to establish the different update and endowment stimuli. The features were: rating an accommodation with stars (5 levels); viewing the average rating; filtering for price, rating etc.; sorting for price, rating etc.; calendar functionality to plan arrival, departure and duration of stay; viewing professional holiday reviews; a budget calculator to find and plan fixed budget vacations. To implement our required assumption regarding an equal value of the employed features (which we raised in our hypotheses), we deliberately defined the task in a way so that it could—in principle—be accomplished without using any of the manipulated features. Nonetheless, each feature made the accomplishment of one part of the task or a specific constraint easier for ‘Tom’, thus providing approximately equal benefits. The vignettes specifically described the usage of each available feature in each condition in order to highlight the benefit of each feature. Regarding the assignment to initial endowment and later update, the order of the seven features was random (as listed above) but held constant across the experimental groups. Group A had all seven features right from the beginning; group B had one feature right from the beginning (rating an accommodation with stars) and the remaining six features were added through an update; group C had the first six features right from the beginning and one feature was added through an update (budget calculator to find and plan fixed budget vacations) and group D had one feature right from the beginning (rating an accommodation with stars) and one feature was added through an update (viewing the average rating). Figure 1 depicts this assignment of features.

To operationalize the manipulation, we constructed textual scenarios and presented them to participants in an online based questionnaire that comprised several consecutive pages. On a first page, we described Tom and his personality. For a high construal level mindset, we described Tom as a person ‘who is considering the big picture for making decisions’, ‘who establishes an overview on superior goals’, ‘who makes gut decisions and focuses on essentials’, for a low construal level mindset we described Tom as a person ‘who wants to consider all details before deciding’, ‘who establishes a broad information basis to consider all facets of a problem’, ‘who wants to decide rational and therefore focuses on details’. Subsequently, we introduced a task of finding a cheap 2-week vacation to Madrid, Spain, with a price limit

¹ 20 subjects participated in the pre-study. They resembled the demographics of the main study and rated 22 identified TBP features with regard to perceived importance on seven-point-Likert scales. Seven features with similar but high levels of importance were selected for the main study. Holding the importance of features constant within and across treatments allowed us to isolate the effects of initial feature endowment and update size on the dependent variables and avoid potential confounding effects that might result from variations in the importance of features.
of 800 Euros, full board, clean restrooms and, among other things, a modern ambience. Third, the software and its initial feature endowment were described as follows: ‘To find a suitable flight and hotel, he [Tom] uses the TBP Journey4You. In addition to the simple search for flights and hotels, the platform offers the following functionalities:’ followed by the abovementioned features. On the second page—part two of the vignette—we first described Tom’s repeated visits of the TBP over several days to find a suitable flight and hotel including the usage of the currently available features. For the three groups which received an update, we subsequently included a section that introduced the update after Tom revisits the TBP one day (halfway through the described usage time) ‘... the website Journey4You shows a message that Journey4You offers new functionality to its users:’ followed by a list of one or six of the features. After this conditional section describing the update, a description of further usage including the use of the new features (only in the update conditions) followed (see Figure 1): ‘Tom uses the new function in addition to available functionality [...]. Finally, Tom locates the most attractive offer and books his journey’. Except for the manipulated text passages, all other parts of the scenario were kept constant across the groups. After this second page, participants started to answer the questionnaire. Participants could only proceed to the next page when all questions were answered and returning to previous pages, including the vignettes, was not possible. Following common vignette procedures (Aguinis and Bradley 2014), we ensured that our vignettes illustrated realistic situations and that participants identified well with the character described by conducting several revision cycles based on qualitative interviews. Furthermore, the so designed vignettes were tested in a pilot study with ten subjects to ensure that our treatments were manipulated according to the experimental design (Perdue and Summers 1986). Specifically, subjects of the pilot study were asked about the comprehensiveness of the instructions, the vignettes’ realism and their ability to put themselves in the hypothetical scenario as well as the clarity of questions in the subsequent questionnaire. Feedback and suggestions were obtained from participants and the vignettes and the questionnaire were accordingly revised for the main experiment.

**Measures**

**Dependent Variables**

For the questionnaire that succeeded our vignettes, we used validated scales to measure dependent variables with slight changes in wording to adapt the items to our experimental setting. Unless stated otherwise all items were measured on seven-point-Likert scales anchored at 1 = strongly disagree and 7 = strongly agree. CI was measured with items adapted from (Hong et al. 2011): CI1. *Tom intends to continue using the TBP rather than discontinue its use;* CI2. *Tom’s intentions are to continue using the TBP than use any alternative means;* DISC was based on items adapted from Bhattacherjee (2001): DISC1. *Tom’s experience with using the TBP were better than what he expected;* DISC2. *The service level provided by the TBP was better than what Tom expected;* DISC3. *Overall, most of Tom’s expectations from using the TBP were confirmed;* While we did not explicitly hypothesize about SAT and PU, we also included these variables in our measurement, to capture the entire continuance model. PU and SAT were measured with items from Kim and Son (2009): PU1. *Using the TBP enhances Tom’s effectiveness;* PU2. *Using the TBP increases Tom’s productivity;* PU3. *Using the TBP improves Tom’s performance;* SAT1. *Tom is content with the features provided by the TBP;* SAT2. *Tom is satisfied with the features provided by the TBP;* SAT3. *What Tom gets from using the TBP meets what he expects for this type of software.*

**Control Variables and Manipulation Check**

In our study, we controlled for subjects’ expertise with regard to TBPs with an established four item, seven-point semantic differential scale with the items *know very little about/know very much about, inexperienced/experienced, uniformed/informed, novice buyer/expert buyer* (Mishra et al. 1993). Furthermore, we also captured participants’ motivation to process information with one item to control for motivational influences on response behavior (Suri and Monroe 2003). Additionally, in the post experimental survey, we measured (1) subjects’ level of understanding of the scenario, (2) the scenario’s realism, (3) subjects’ ability to put themselves in the hypothetical setting described in the scenario, (4) subjects’ level of understanding of the instructions and questionnaire items. We further collected the participants’ age, gender, education and profession to control for a homogeneous distribution of participants across groups with regard to these core demographics.
Participants, Incentives and Procedures

The outlined online experiment was conducted between November 2014 and January 2015. In line with best-practices of augmented number and diversity of participants for vignette studies (Aguinis and Bradley 2014), we invited subjects to participate in an online survey about software usage by consumers through several postings in social networks, via word-of-mouth and emails. Overall, 386 subjects started the experiment. The rate of completion was 74%, i.e., a total number of 285 subjects completed the questionnaire. 24 participants were excluded from our final analysis because they either did not answer control questions correctly or completed the experiment in less than five minutes (the average time to completion was about ten minutes). Of the 261 remaining participants that were used in the following analysis, 107 were females and 153 were males (one not specified). Subjects’ average age was 28.47 (SD=9.08) years. On average, 78% of the subjects used TBPs less than one hour per month, 20% one to five hours and 2% more than five hours per month. The average reported expertise with a TBP was 4.26 (SD=1.63) on a seven-point semantic differential scale. 48% of subjects were students, 29% employees, 8% self-employed and the remainder had various occupations.

Data Analysis and Results

Control Variables and Manipulation Check

To confirm a successful randomization of participants to the different experimental conditions, we searched for differences between groups with regard to the collected control variables by performing a one-way MANOVA. The results showed no significant differences between groups $\lambda = 0.90$, $F[33,717]=0.78$, $p>0.1$. Neither control variable showed significant differences: age ($F=0.72$, $df=3$, $p>0.1$), gender ($F=1.40$, $df=3$, $p>0.1$), occupation ($F=1.38$, $df=3$, $p>0.1$), usage intensity ($F=0.54$, $df=3$, $p>0.1$), product expertise ($F=0.91$, $df=3$, $p>0.1$), motivation to process information ($F=1.21$, $df=3$, $p>0.1$), understanding of story ($F=0.24$, $df=3$, $p>0.1$), story’s realism ($F=1.31$, $df=3$, $p>0.1$), ability to put oneself in the scenario ($F=0.14$, $df=3$, $p>0.1$), understanding questions ($F=0.23$, $df=3$, $p>0.1$) and instructions ($F=0.17$, $df=3$, $p>0.1$). Hence, we conclude that subjects were distributed homogenously across our experimental groups.

As indicators for the external validity of our findings, we reviewed the participants’ answers regarding their understanding, realism and adaption of the vignettes. For all three measures, the participants reported high levels on the seven-point-Likert-scales: understanding ($M=6.32$; $SD=1.00$), realism ($M=5.84$; $SD=1.31$) and adaption ($M=5.77$; $SD=1.36$). Moreover, in qualitative open text questions we observed that subjects described Tom’s instead of their own feelings and thoughts. It is therefore reasonable to assume that our experimental manipulations using textual vignettes worked as intended and that participants thought and acted like the fictitious character. To control for potential differences in the effect of updates on CI from different construal levels, we ran two one-way ANOVA tests. We assessed whether there were any differences in CI between high and low construal conditions across the control group and three update groups. The results indicated no significant differences for CI in the control group ($F=1.16$, $df=1$, $p>0.1$) and all three update groups ($F=0.06$, $df=1$, $p>0.1$). We may therefore conclude that construal level did not interact with the effect of feature updates on CI. In our subsequent analysis we thus combined participants who received high and low construal level treatments.

Measurement Validation

Because we adopted established constructs for our measurement, confirmatory factor analysis (CFA) was conducted to test the instruments’ convergent and discriminant validity (Levine 2005). Table 1 reports the CFA results regarding convergent validity using SmartPLS 3.0 (Chin et al. 2003; Ringle et al. 2014).
Table 1. Results of Confirmatory Factor Analysis for Core Variables

<table>
<thead>
<tr>
<th>Latent Construct</th>
<th>Items</th>
<th>Range of std. Factor Loadings*</th>
<th>Cronbach’s alpha</th>
<th>Composite Reliability (ρc)</th>
<th>Average Variance Extracted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuance intention (CI)</td>
<td>2</td>
<td>0.925 - 0.929</td>
<td>0.836</td>
<td>0.924</td>
<td>0.859</td>
</tr>
<tr>
<td>Disconfirmation (DISC)</td>
<td>3</td>
<td>0.779 - 0.879</td>
<td>0.793</td>
<td>0.879</td>
<td>0.708</td>
</tr>
<tr>
<td>Perceived Usefulness (PU)</td>
<td>3</td>
<td>0.783 - 0.870</td>
<td>0.796</td>
<td>0.879</td>
<td>0.709</td>
</tr>
<tr>
<td>Satisfaction (SAT)</td>
<td>3</td>
<td>0.772 - 0.920</td>
<td>0.840</td>
<td>0.905</td>
<td>0.761</td>
</tr>
</tbody>
</table>

* All factor loadings are significant at the p<0.01 level

Constructs were assessed for reliability using Cronbach's alpha (Cronbach 1951). Values above 0.70 are considered to provide adequate reliability (Nunnally 1994). The alphas for all constructs were well above 0.7. Moreover, the composite reliability of all constructs exceeded 0.70, which is considered the minimum threshold (Hair et al. 2011). Values for AVEs for each construct ranged from 0.708 to 0.859, exceeding the variance due to measurement error for that construct (that is, AVE exceeded 0.50). Moreover, we examined cross correlations (see Table 2). All square roots of AVE exceeded inter-construct correlations, providing strong evidence of discriminant validity. Hence, the constructs in our study represent concepts that are both theoretically and empirically distinguishable. After ensuring the validity of our measured constructs, summed scales based on the average scores of the multi-items were used to calculate the constructs for our later analysis (Zhu et al. 2012).

Table 2. Means, Standard Deviations and Correlation Matrix for Core Variables

<table>
<thead>
<tr>
<th>Latent Construct</th>
<th>M</th>
<th>SD</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Continuance intention (CI)</td>
<td>5.644</td>
<td>1.183</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2) Disconfirmation (DISC)</td>
<td>5.450</td>
<td>1.028</td>
<td>0.458***</td>
<td>0.841</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3) Perceived Usefulness (PU)</td>
<td>5.393</td>
<td>1.159</td>
<td>0.484***</td>
<td>0.549***</td>
<td>0.842</td>
<td></td>
</tr>
<tr>
<td>(4) Satisfaction (SAT)</td>
<td>5.865</td>
<td>0.973</td>
<td>0.478***</td>
<td>0.654***</td>
<td>0.608***</td>
<td>0.872</td>
</tr>
</tbody>
</table>

Note: Bolded diagonal elements are the square root of AVE. These values should exceed inter-construct correlations (off-diagonal elements) for adequate discriminant validity; ***p<0.01, **p<0.05, *p<0.1.

Hypotheses Testing

In order to test our hypotheses H1 - H3, we conducted a one-way ANOVA with planned contrast tests using IBM SPSS Statistics 22. A significant effect of feature updates on CI was found (F=8.362, p<0.01). Post hoc contrast analysis revealed that participants in all three update groups (B, C and D) showed significant higher levels of CI compared to the control group (see Figure 2).
When Updates Make a User Stick

In hypothesis H1, we argued that software that receives additional functionality via feature updates will induce higher CI compared to software that includes all these features right with the first release. The results from our experiment indicate that participants’ CI in group B (one initial feature, update adds six features) was on average significantly higher (+0.77) than in group A (seven initial features, no updates). Hence, H1 is supported. Hypothesis H2 posits that this effect would persist, regardless of the initial feature endowment. As hypothesized, our results showed that participants’ CI in group C (six initial features, update adds one feature) was on average significantly higher than in group A (+0.54). Furthermore, H2 implies that the increase in user’s CI should not be lower when an update is applied to software with high initial feature endowment (compared to software with low initial feature endowment). However, the results from our experiment indicated that this idea is not supported, since subject’s CI in group C was on average significantly lower (-0.40) than in group D (one initial feature, update adds one feature). H2 is thus only supported partly. A discussion of this finding follows in the next section. In hypothesis H3 we proposed that the positive effect of feature updates on user’s CI (compared to software that includes all these features right from the first release) persists regardless of the update size. Supporting this hypothesis, our results showed that subject’s CI in group D was on average significantly higher (+0.94) than in the control group A. Moreover, hypothesis H3 also implies that the increase in user’s CI should not be higher for a feature update with large size compared to a small update size. Since subject’s CI in group D (one initial feature, update adds on feature) was not significantly different than in group B (one initial feature, update adds six features), we conclude that H3 is fully supported.

Finally, to evaluate the explanatory mechanism behind the impact of feature updates on CI, we conducted a mediation analysis using partial least squares structural equation modeling with SmartPLS 3.0 with the bootstrapping resampling procedure (Ringle et al. 2014). In line with previous post-adoption continuance studies (Bhattacherjee 2001; Bhattacherjee and Premkumar 2004; Kim and Son 2009; Ortiz de Guinea and Webster 2013), a component based structural equation modeling approach using SmartPLS was preferred over a covariance-based one because it does not impose sample size restrictions or require multivariate normality distributions of the underlying data. A complete bootstrapping with 10,000 samples was conducted, bias-correction was enabled and test type was set to two tailed. The validation of the employed reflective measurement model is reported in Tables 1 and 2 (Chin et al. 2003). Following hypothesis H4, we included our experimental treatment (no update vs. feature update) as dichotomous independent variable and driver of DISC in the continuance model (Bhattacherjee 2001) (see Figure 3).

Figure 2. Average Continuance Intentions, Mean Differences and Significance Levels

In hypothesis H1, we argued that software that receives additional functionality via feature updates will induce higher CI compared to software that includes all these features right with the first release. The results from our experiment indicate that participants’ CI in group B (one initial feature, update adds six features) was on average significantly higher (+0.77) than in group A (seven initial features, no updates). Hence, H1 is supported. Hypothesis H2 posits that this effect would persist, regardless of the initial feature endowment. As hypothesized, our results showed that participants’ CI in group C (six initial features, update adds one feature) was on average significantly higher than in group A (+0.54). Furthermore, H2 implies that the increase in user’s CI should not be lower when an update is applied to software with high initial feature endowment (compared to software with low initial feature endowment). However, the results from our experiment indicated that this idea is not supported, since subject’s CI in group C was on average significantly lower (-0.40) than in group D (one initial feature, update adds one feature). H2 is thus only supported partly. A discussion of this finding follows in the next section. In hypothesis H3 we proposed that the positive effect of feature updates on user’s CI (compared to software that includes all these features right from the first release) persists regardless of the update size. Supporting this hypothesis, our results showed that subject’s CI in group D was on average significantly higher (+0.94) than in the control group A. Moreover, hypothesis H3 also implies that the increase in user’s CI should not be higher for a feature update with large size compared to a small update size. Since subject’s CI in group D (one initial feature, update adds on feature) was not significantly different than in group B (one initial feature, update adds six features), we conclude that H3 is fully supported.

Finally, to evaluate the explanatory mechanism behind the impact of feature updates on CI, we conducted a mediation analysis using partial least squares structural equation modeling with SmartPLS 3.0 with the bootstrapping resampling procedure (Ringle et al. 2014). In line with previous post-adoption continuance studies (Bhattacherjee 2001; Bhattacherjee and Premkumar 2004; Kim and Son 2009; Ortiz de Guinea and Webster 2013), a component based structural equation modeling approach using SmartPLS was preferred over a covariance-based one because it does not impose sample size restrictions or require multivariate normality distributions of the underlying data. A complete bootstrapping with 10,000 samples was conducted, bias-correction was enabled and test type was set to two tailed. The validation of the employed reflective measurement model is reported in Tables 1 and 2 (Chin et al. 2003). Following hypothesis H4, we included our experimental treatment (no update vs. feature update) as dichotomous independent variable and driver of DISC in the continuance model (Bhattacherjee 2001) (see Figure 3).
When Updates Make a User Stick

Following Hair et al. (2014), our analysis revealed significant paths between all core variables of the continuance model (Bhattacherjee 2001). Overall, the continuance model was strongly supported and explained about 28% of the variance in continuance intention ($R^2=0.282$). Moreover, the positive and highly significant path from feature update to DISC ($\beta = 0.17, p<0.01$), supports our hypothesis H4 which suggested that the positive effect from feature updates on CI is partially carried over by DISC to the downstream factors of the IS continuance model to affect CI. In order to further examine the mediation by disconfirmation, additional models were tested by including direct links from feature update to continuance intention (Venkatesh 2000). The effect of feature update on continuance intention was partially mediated by disconfirmation (and the subsequent variables perceived ease of use and satisfaction) (Hair 2014).

**Discussion**

This study sought to achieve three main objectives: (1) examine the effects of feature updates on users’ intentions to continue using an information system (i.e., whether there is a discernible effect from updates), (2) to investigate two possible boundary conditions (i.e., when there is an effect from updates and when not), namely, update size and initial feature endowment and (3) to unravel the explanatory mechanism through which such an effect occurs (i.e., how and why such an effect from updates occurs). To achieve these objectives, we drew on the IS continuance model that is embedded in the expectation-confirmation theory and investigated our hypotheses based on a vignette-based online experiment.

Drawing on the advantages of the experimental method, which allows to isolate the effects of manipulated stimuli on user responses from other confounding variables to unveil causal relationships, we found that CI was rated significantly higher in all update-conditions (groups B, C and D) than in the non-update condition (A). This increase in CI can be interpreted as a somewhat counter-intuitive finding because in the groups in which features were delivered via updates only halfway through the described time span, users had access to less functionality to accomplish their task compared to the user who had all features right at the beginning: They had to use the software prior to the update with less features. In particular in group D, even after the update, the user had in sum fewer features to accomplish his task compared to group A. Despite this objective disadvantage, participants in all update groups, including D, indicated significantly higher scores in CI. This suggests the presence of a positive, somewhat non-rational user response to feature updates and challenges the idea of a ‘rational user’ in the IS continuance literature (Bhattacherjee and Barfar 2011; Ortiz de Guinea and Markus 2009; Ortiz de Guinea and Webster 2013).

Moreover, our experiment revealed that update size does not seem to constitute a boundary condition to this positive effect of feature updates on users’ CI (groups B and D did not differ significantly). However, contrary to our hypothesis and to what the continuance model and the underlying ECT mechanism would suggest regarding the subjective and relative evaluation of feature endowment and updates, groups C and D differed significantly with regard to CI. Feature endowment therefore appears to moderate the effect of feature updates on CI. While this finding partly contradicts our second hypothesis, the observed effect may be explained by the concept of diminishing sensitivity (Tversky and Kahneman 1992). This concept suggests that the characteristics of a product to which a new feature is added determine the impact of this feature on e.g., the sales of the product. Specifically, a feature that is added to a relatively superior product
increases the overall perceived value of the product less than the same feature that is added to a relatively inferior product (Nowlis and Simonson 1996). Nonetheless, we suggest further research to substantiate our interpretation of this finding. Additionally, we could demonstrate that the positive effect of feature updates on CI was mediated by a serial effect chain of relations that originates in a positive disconfirmation of previous expectations (DISC). This emphasizes the relative nature of users’ evaluations of changes in software features and validates the IS continuance model for IS that are conceived as a dynamic collection of features rather than one monolithic and static block. These changes in beliefs and attitudes over time which are induced by changes in the IT artifact may be explained by sequential belief updating (Kim 2009; Maier et al. 2015). Lastly, our experiment revealed that these findings even persist under different user’s construal levels, which are modes of thinking that seemed likely to affect the perception of the allocation of features between initial release and later updates.

**Implications for Research**

This study makes three contributions to literature. First, our main contribution lies in the detection of a positive user reaction to feature updates. Specifically, delivering software functionality through feature updates has a stronger and more positive impact on IS users’ CI than providing the entire feature set at once and right with the first release. We reveal that users evaluate software functionality not objectively and that evaluations of feature updates are based on relative comparisons to a subjective baseline of functionality. While this effect persists despite a high initial feature endowment, its magnitude diminishes with increasing endowment. This diminishing sensitivity to new features is consistent with findings from psychology and marketing research (Nowlis and Simonson 1996) and should be considered when studying users’ perceptions of software features. With regard to update size, we find that the positive effect of feature updates is independent from different update sizes. This implies that users do not assess changes in an information system through updates objectively. Moreover, in our study we could empirically demonstrate that the observed effects even withstand different construal levels, a crucial user characteristic with respect to preferences of initial over later benefits. The lack of a significant influence of construal level further substantiates the robustness of our findings. Our second contribution lies in shedding light on the explanatory mechanism behind the identified positive effect of feature updates on CI. In particular, we find that this positive effect is mediated by a serial chain of relations which originates in the positive disconfirmation of previous expectations. This finding highlights the subjective and relative nature of users’ perceptions of IS changes which lead to somewhat non-rational responses (Fleischmann et al. 2014). These results may also be interpreted as a possible empirical evidence for reference point dependency in users’ perception of software (Kahneman and Tversky 1979) and deserve further research. Our third and overarching contribution lies in the extension of the predominant view of information systems in the post-adoptions literature from a mostly monolithic one to a finer-grained and dynamic perspective by showing how modular features can be strategically combined by vendors and that the specific composition of features and their changes over time can influence users’ beliefs and attitudes regarding a software. In doing so, we answer calls of several IS scholars (e.g., Benbasat and Barki 2007; Jasperson et al. 2005) to consider the granularity of information systems in IS research. Our study thus offers a complement to the existing IS post-adoptions literature by showing that user beliefs and attitudes change alongside modifications of the IT artifact during usage. Moreover, through this notion, our study also contributes to the stream of IS research on belief updating (Kim 2009; Maier et al. 2015).

**Implications for Practice**

Our results also have important implications for practice. First, despite the extensive use of feature updates by vendors to maintain, alter and extend their products after they have already been rolled out, it is surprising that insights on how these updates are perceived by users are still scarce. This leaves vendors without guidance when to provide which functionality to customers. From the results of our experimental study we can conclude that it might be advisable for vendors to deliver features over time, via updates, because feature updates can induce a positive experience, which, in turn, increases users’ CI. For vendors, users with a high CI are a particularly desirable goal because these are the loyal, returning customers who ensure the long term profitability of their businesses in the highly competitive software industry. Moreover, a high CI is particularly important for the increasing share of subscription-based business models in the software industry (Veit et al. 2014). An incremental delivery of features may also be beneficial to vendors, providing them with a competitive advantage due to shorter times-to-market when
When Updates Make a User Stick

developing new software. Instead of waiting for the completion of all planned features, they could release their software with a smaller feature endowment and deliver additional functionalities successively through incremental feature updates when their development is completed. An additional benefit of this quicker time-to-market strategy is that revenues start to flow earlier than under an all-at-once feature delivery strategy with a later release. However, the identified positive effect of feature updates needs to be well understood and correctly applied to achieve the desired outcomes. The findings of this study reveal that this effect works only if users’ experiences exceed their prior expectations when receiving an update (positive disconfirmation). Vendors should thus avoid announcing feature updates in advance as this would annihillate the required element of surprise. Our results regarding initial feature endowment show that while this positive effect from updates decreases with endowment it still persists. A deferred delivery of features as suggested by our hypothesis H1 may thus be applied for lean software as well as for software with a high initial feature endowment. Vendors of advanced, mature software may therefore also take advantage of this effect. Furthermore, because the size of updates was revealed to not affect this positive effect, it is not necessary for vendors to pack several features into one update in order to obtain this positive user response. However, vendors should not overdraw holding back functionality. Starting out with a too small feature set might render the first release of a software almost useless and lead to discontinuation before the program can be updated or even prohibit the adoption in the first place. Finally, our findings highlight the benefit from using a modular software-architecture. Aside from an increased flexibility in the development and maintenance, a modular architecture also facilitates the use of feature updates. When features are encapsulated in discrete modules, they may be delivered in small packages (updates) and can be integrated easily in existing systems that are already being used.

Limitations and Future Research

Five limitations of this study are noteworthy and provide avenues for future research. First, this study made some crucial assumptions that can be revisited in future research. We conceptualized individual features to provide equal value to the user and thus held the relative importance of features constant. This was also reflected in the design of our experiment. To increase the external validity of our findings, future studies should investigate features with varying relative importance and account for different valuations of features across users. Second, our treatment was realized through vignettes in an online questionnaire. As such, typical limitations of this methodology apply (Aguinis and Bradley 2014): the setting was fictitious and demanded subjects to put themselves in the position of the person in the scenario, while no instructor was available if questions arose. We thus controlled for motivation to process information, understanding and realism of the scenario and have strong reason to rule out bias in our results from these limitations. Nevertheless, we encourage researchers to conduct longitudinal field studies or experiments with real software usage to further validate our findings. Third, we only observed a short usage time span and one update in our experiment. Future studies could explore user responses to repeated updates to understand the interplay between update size and update frequency as other possible boundary conditions and thus also deepen the understanding of sequential belief updating triggered by feature updates. Experiments conducted on longer time spans with users’ evaluations measured at several points in time could also provide additional evidence for the robustness of the positive effect of feature updates on users’ CI. Fourth, to control for the potential impact of different construal levels on the perception of updates, we split our experimental groups, resulting in reduced cell sizes for analysis. Although group sizes remained sufficiently large for our thorough statistical analysis and were in line with other, comparable experimental IS studies (e.g., Hong et al. 2004) we encourage future research to increase sample size. Moreover, future research should also explore additional crucial control variables related to users’ short term interests, such as propensity to resist change (Oreg 2003; Polites and Karahanna 2012), stress (Maier et al. 2015) or habit (Polites and Karahanna 2013). Fifth, the positive effect of feature updates on users’ CI was shown to work for an online service. However, the domain of online services the technical complexity of the update process and potential downsides in the user experience are largely hidden from the user. While we believe that this unobtrusiveness of updates applies to a wide and also increasing range of modern software products and services (web services, platforms for mobile devices and modern desktop operating systems) there might be types of software for which our results are not generalizable (e.g., legacy software). Future research is encouraged to show the same effect for other types of software.
Conclusion

Feature updates have become a pervasively used instrument of software vendors to maintain, alter, and extend their products over time. Despite their prevalence in private and business IT usage contexts, their effects on crucial user reactions in the IS post-adoption context have remained largely unexplored. This study is among the first to demonstrate that feature updates have the potential to increase users' CI above and beyond a level generated by monolithic software packages that deliver the entire feature set at once. It also reveals the robustness of this effect by ruling out update size and users' construal level as potential boundary conditions to this phenomenon. Nonetheless, we identified users' valuations of feature updates to slightly diminish with increasing feature endowment of the updated software. Lastly, this study identified a positive disconfirmation of previous expectations as the underlying mechanism by which feature updates influence users' CI. In summary, this study represents an important first step towards a better understanding of the nature of feature updates and how they affect user reactions. It may therefore serve as a springboard for future studies on feature updates in the IS post-adoption context.

References

When Updates Make a User Stick


