Practical Guide to Controlled Experiments on the Web: Listen to Your Customers not to the HiPPO

Ron Kohavi
Microsoft
One Microsoft Way
Redmond, WA 98052
ronnyk@microsoft.com

Randal M. Henne
Microsoft
One Microsoft Way
Redmond, WA 98052
rhenne@microsoft.com

Dan Sommerfield
Microsoft
One Microsoft Way
Redmond, WA 98052
dans@microsoft.com

ABSTRACT

The web provides an unprecedented opportunity to evaluate ideas quickly using controlled experiments, also called randomized experiments (single-factor or factorial designs), A/B tests (and their generalizations), split tests, Control/Treatment tests, and parallel flights. Controlled experiments embody the best scientific design for establishing a causal relationship between changes and their influence on user-observable behavior. We provide a practical guide to conducting online experiments, where end-users can help guide the development of features. Our experience indicates that significant learning and return-on-investment (ROI) are seen when development teams listen to their customers, not to the Highest Paid Person's Opinion (HiPPO). We provide several examples of controlled experiments with surprising results. We review the important ingredients of running controlled experiments, and discuss their limitations (both technical and organizational). We focus on several areas that are critical to experimentation, including statistical power, sample size, and techniques for variance reduction. We describe common architectures for experimentation systems and analyze their advantages and disadvantages. We evaluate randomization and hashing techniques, which we show are not as simple in practice as is often assumed. Controlled experiments typically generate large amounts of data, which can be analyzed using data mining techniques to gain deeper understanding of the factors influencing the outcome of interest, leading to new hypotheses and creating a virtuous cycle of improvements. Organizations that embrace controlled experiments with clear evaluation criteria can evolve their systems with automated optimizations and real-time analyses. Based on our extensive practical experience with multiple systems and organizations, we share key lessons that will help practitioners in running trustworthy controlled experiments.

Categories and Subject Descriptors

G.3 **Probability and Statistics/Experimental Design**: controlled experiments, randomized experiments, A/B testing.
I.2.6 **Learning**: real-time, automation, causality.

General Terms

Management, Measurement, Design, Experimentation, Human Factors.

Keywords

Controlled experiments, A/B testing, e-commerce.

1. INTRODUCTION

One accurate measurement is worth more than a thousand expert opinions
— Admiral Grace Hopper

In the 1700s, a British ship's captain observed the lack of scurvy among sailors serving on the naval ships of Mediterranean countries, where citrus fruit was part of their rations. He then gave half his crew limes (the Treatment group) while the other half (the Control group) continued with their regular diet. Despite much grumbling among the crew in the Treatment group, the experiment was a success, showing that consuming limes prevented scurvy. While the captain did not realize that scurvy is a consequence of vitamin C deficiency, and that limes are rich in vitamin C, the intervention worked. British sailors eventually were compelled to consume citrus fruit regularly, a practice that gave rise to the still-popular label *limeys* (1).

Some 300 years later, Greg Linden at Amazon created a prototype to show personalized recommendations based on items in the shopping cart (2). You add an item, recommendations show up; add another item, different recommendations show up. Linden notes that while the prototype looked promising, "a marketing senior vice-president was dead set against it," claiming it will distract people from checking out. Greg was "forbidden to work on this any further." Nonetheless, Greg ran a controlled experiment, and the "feature won by such a wide margin that not having it live was costing Amazon a noticeable chunk of change. With new urgency, shopping cart recommendations launched." Since then, multiple sites have copied cart recommendations.

The authors of this paper were involved in many experiments at Amazon, Microsoft, Dupont, and NASA. The culture of experimentation at Amazon, where data trumps intuition (3), and a system that made running experiments easy, allowed Amazon to innovate quickly and effectively. At Microsoft, there are multiple systems for running controlled experiments. We describe several architectures in this paper with their advantages and disadvantages. A unifying theme is that controlled experiments have great return-on-investment (ROI) and that building the appropriate infrastructure can accelerate innovation. Stefan Thomke's book title is well suited here: **Experimentation Matters** (4).

The web provides an unprecedented opportunity to evaluate ideas quickly using controlled experiments, also called randomized experiments (single-factor or factorial designs), A/B tests (and their generalizations), split tests, Control/Treatment, and parallel flights. In the simplest manifestation of such experiments, live users are randomly assigned to one of two variants: (i) the Control, which is commonly the "existing" version, and (ii) the Treatment, which is usually a new version being evaluated.

Metrics of interest, ranging from runtime performance to implicit and explicit user behaviors and survey data, are collected. Statistical tests are then conducted on the collected data to evaluate whether there is a statistically significant difference between the two variants on metrics of interest, thus permitting us to retain or reject the (null) hypothesis that there is no difference between the versions. In many cases, drilling down to segments of users using manual (e.g., OLAP) or machine learning and data mining techniques, allows us to understand which subpopulations show significant differences, thus helping improve our understanding and progress forward with an idea.

Controlled experiments provide a methodology to reliably evaluate ideas. Unlike other methodologies, such as post-hoc analysis or interrupted time series (quasi experimentation) (5), this experimental design methodology tests for causal relationships (6 pp. 5-6). Most organizations have many ideas, but the return-on-investment (ROI) for many may be unclear and the evaluation itself may be expensive. As shown in the next section, even minor changes can make a big difference, and often in unexpected ways. A live experiment goes a long way in providing guidance as to the value of the idea. Our contributions include the following.

- In Section 3 we review controlled experiments in a web environment and provide a rich set of references, including an important review of statistical power and sample size, which are often missing in primers. We then look at techniques for reducing variance that we found useful in practice. We also discuss extensions and limitations so that practitioners can avoid pitfalls.
- In Section 4 we present generalized architectures that unify multiple experimentation systems we have seen, and we discuss their pros and cons. We show that some randomization and hashing schemes fail conditional independence tests required for statistical validity.
- In Section 5 we provide important practical lessons.

When a company builds a system for experimentation, the cost of testing and experimental failure becomes small, thus encouraging innovation through experimentation. Failing fast and knowing that an idea is not as great as was previously thought helps provide necessary course adjustments so that other more successful ideas can be proposed and implemented.

2. MOTIVATING EXAMPLES

The fewer the facts, the stronger the opinion
— Arnold Glasow

The following two examples show how small differences in UI can result in significant differences to the metric of interest.

2.1 Checkout Page at Doctor FootCare

The *conversion rate* of an e-commerce site is the percentage of visits to the website that include a purchase. The following example comes from Bryan Eisenberg's articles (7; 8).

There are nine differences between the two variants of the Doctor FootCare checkout page shown in Figure 1. If a designer showed you these and asked which one should be deployed, could you tell which one results in a higher conversion rate? Could you estimate what the difference is between the conversion rates and whether that difference is significant?

We will share the results at the end of the section, but we encourage the readers to think about it before reading the answer.

2.2 Ratings of Microsoft Office Help Articles

Users of Microsoft Office who request help from Office or through the website are given an opportunity to rate the article. The initial implementation presented users with a *Yes/No* widget. The team then modified the widget and offered a 5-star ratings.

The motivations for the change were the following:

1. The 5-star widget provides finer-grained feedback, which might help better evaluate content writers.
2. The 5-star widget improves usability by exposing users to a single feedback box as opposed to two separate pop-ups (one for *Yes/No* and another for *Why*).

We encourage you, the reader, to think about whether the new model can meet the stated goals.

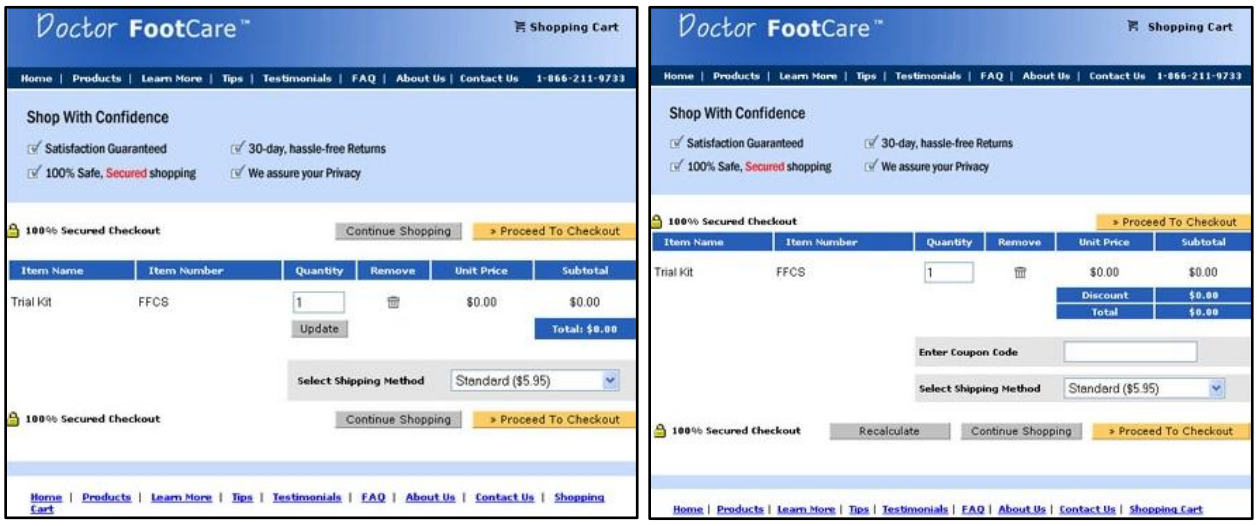

Figure 1: Variant A on left, Variant B on right.
Can you guess which one has a higher conversion rate and whether the difference is significant?

2.3 Results and ROI

For the Doctor FootCare checkout page, variant A in Figure 1 outperformed variant B by an order of magnitude. In reality, the site "upgraded" from the A to B and lost 90% of their revenue. Most of the changes in the upgrade were positive, but the coupon code was the critical one: people started to think twice about whether they were paying too much because there are discount coupons out there that they do not have. By removing the discount code from the new version (B), conversion-rate increased 6.5% relative to the old version (A).

For Microsoft's Office Help articles, the number of ratings plummeted by an order of magnitude, thus significantly missing on goal #2 above. Based on additional tests, it turned out that the two-stage model actually helps in increasing the response rate. Even goal #1 was somewhat of a disappointment as most people chose the extremes (one or five stars). When faced with a problem for which you need help, the article either helps you solve the problem or it does not!

While these are extreme examples that are surprising in the magnitude of the difference, they show how hard it is to predict the success of new designs. Both of these are user-interface examples, but controlled experiments can be used heavily in back-end algorithms (e.g., recommendations, search relevance, etc).

Great examples of experiments are available at Marketing Experiments journal (9), Design Choices Can Cripple a Website (10), Call to Action (11), and Which Sells Best (12). Forrester's Primer on A/B Testing (13) mentions a few good examples of positive ROI. In shop.com's The State of Retailing (14), the authors wrote that in their survey of 137 US retailers "100% of the retailers that employed usability testing and A/B testing of offers and promotions rank these tactics as effective or very effective."

3. CONTROLLED EXPERIMENTS

Enlightened trial and error outperforms the planning of flawless execution
— David Kelly, founder of Ideo

In the simplest controlled experiment, often referred to as an A/B test, users are randomly exposed to one of two variants: control (A), or treatment (B) as shown in Figure 2 (15; 16; 6).

The key here is "random." Users cannot be distributed "any old which way" **(17)**; no factor can influence the decision. Based on observations collected, an Overall Evaluation Criterion (OEC) is derived for each variant **(18)**.

For example, in Checkout Example (Section 2.1), the OEC can be the conversion rate, units purchased, revenue, profit, expected lifetime value, or or some weighted combination of these. Analysis is then done to determine if the difference in the OEC for the variants is statistically significant.

If the experiment was designed and executed properly, the only thing consistently different between the two variants is the change between the Control and Treatment, so any differences in the OEC are inevitably the result of this assignment, establishing causality **(17 p. 215)**.

There are several primers on running controlled experiments on the web **(19 pp. 76-78; 11 pp. 283-286; 13; 20; 21; 22; 23; 24)**, (25; 26 pp. 248-253; 27 pp. 213-219; 28 pp. 116-119).

While the concept is easy to understand and basic ideas echo through many references, there are important lessons that we share here that are rarely discussed. These will help experimenters understand the applicability, limitations, and how to avoid mistakes that invalidate the results.

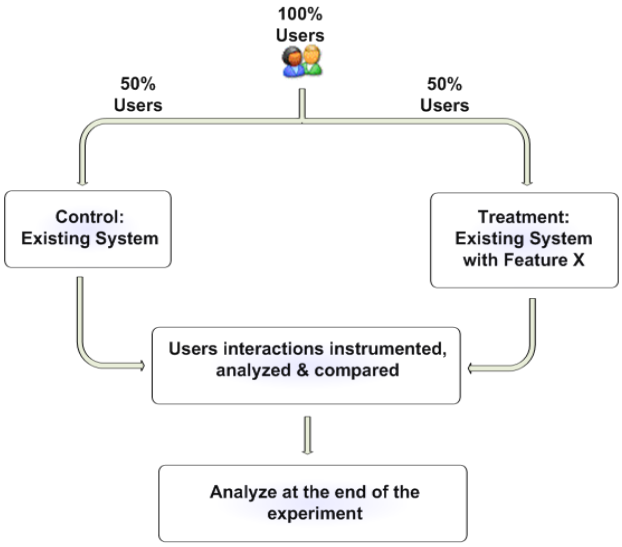

Figure 2

3.1 Terminology

The terminology for controlled experiments varies widely in the literature. Below we define key terms used in this paper and note alternative terms that are commonly used.

Overall Evaluation Criterion (OEC) (18). A quantitative measure of the experiment's objective. In statistics this is often called the **Response** or **Dependent Variable** (15; 16); other synonyms include **Outcome**, **Evaluation metric**, **Performance metric**, or **Fitness Function** (22). Experiments may have multiple objectives and a scorecard approach might be taken (29), although selecting a single metric, possibly as a weighted combination of such objectives is highly desired and recommended (18 p. 50). A single metric forces tradeoffs to be made once for multiple experiments and aligns the organization behind a clear objective. A good OEC should not be short-term focused (e.g., clicks); to the contrary, it should include factors that predict long-term goals, such as predicted lifetime value and repeat visits. Ulwick describes some ways to measure what customers want (although not specifically for the web) (30).

Factor. A controllable experimental variable that is thought to influence the OEC. Factors are assigned **Values,** sometimes called **Levels** or **Versions.** Factors are sometimes called **Variables.** In simple A/B tests, there is a single factor with two values: A and B.

Variant. A user experience being tested by assigning levels to the factors; it is either the Control or one of the Treatments. Sometimes referred to as **Treatment**, although we prefer to specifically differentiate between the Control, which is a special variant that designates the existing version being compared against and the new Treatments being tried. In case of a bug, for example, the experiment is aborted and all users should see the Control variant.

Experimentation Unit. The entity on which observations are made. Sometimes called an **item**. The units are assumed to be

independent. In the web, the user is the most common experimentation unit, although some experiments may be done on sessions or page views. For the rest of the paper, we will assume that the experimentation unit is a user. It is important that the user receive a consistent experience throughout the experiment, and this is commonly achieved through cookies.

Null Hypothesis. The hypothesis, often referred to as H_0, that the OECs for the variants are not different and that any observed differences during the experiment are due to random fluctuations.

Confidence level. The probability of failing to reject (i.e., retaining) the null hypothesis when it is true.

Power. The probability of correctly rejecting the null hypothesis, H_0, when it is false. Power measures our ability to detect a difference when it indeed exists.

A/A Test. Sometimes called a Null Test (19). Instead of an A/B test, you exercise the experimentation system, assigning users to one of two groups, but expose them to exactly the same experience. An A/A test can be used to (i) collect data and assess its variability for power calculations, and (ii) test the experimentation system (the Null hypothesis should be rejected about 5% of the time when a 95% confidence level is used).

Standard Deviation (Std-Dev). A measure of variability, typically denoted by σ.

Standard Error (Std-Err). For a statistic, it is the standard deviation of the sampling distribution of the sample statistic (15). For a mean of n independent observations, it is $\hat{\sigma}/\sqrt{n}$ where $\hat{\sigma}$ is the estimated standard deviation.

3.2 Hypothesis Testing and Sample Size

To evaluate whether one of the treatments is different than the Control, a statistical test can be done. We accept a Treatment as being statistically significantly different if the test rejects the null hypothesis, which is that the OECs are not different.

We will not review the details of the statistical tests, as they are described very well in many statistical books (15; 16; 6).

What is important is to review the factors that impact the test:

1. **Confidence level.** Commonly set to 95%, this level implies that 5% of the time we will incorrectly conclude that there is a difference when there is none (Type I error). All else being equal, increasing this level reduces our power (below).
2. **Power.** Commonly desired to be around 80-95%, although not directly controlled. If the Null Hypothesis is false, i.e., there is a difference in the OECs, the power is the probability of determining that the difference is statistically significant. (A Type II error is one where we retain the Null Hypothesis when it is false.)
3. **Standard Error.** The smaller the Std-Err, the more powerful the test. There are three useful ways to reduce the Std-Err:
   a. The estimated OEC is typically a mean of large samples. As shown in Section 3.1, the Std-Err of a mean decreases proportionally to the square root of the sample size, so increasing the sample size, which usually implies running the experiment longer, reduces the Std-Err and hence increases the power.
   b. Use OEC components that have inherently lower variability, i.e., the Std-Dev, σ, is smaller. For example, conversion probability (0-100%) typically has lower Std-Dev than number of purchase units (typically small integers), which in turn has a lower Std-Dev than revenue (real-valued).
   c. Lower the variability of the OEC by filtering out users who were not exposed to the variants, yet were still included in the OEC. For example, if you make a change to the checkout page, analyze only users who got to the page, as everyone else adds noise, increasing the variability.
4. The effect, or the difference in OECs for the variants. Larger differences are easier to detect, so great ideas will unlikely be missed. Conversely, if Type I or Type II errors are made, they are more likely when the effects are small.

The following formula approximates the desired sample size, assuming the desired confidence level is 95% and the desired power is 90% (31):

$$n = (4r\sigma/\Delta)\text{^}2$$

where n is the sample size, r is the number of variants (assumed to be approximately equal in size), σ is the std-dev of the OEC, and Δ is the minimum difference between the OECs. The factor of 4 may overestimate by 25% for large n (32; 33), but the approximation suffices for the example below.

Suppose you have an e-commerce site and 5% of users who visit during the experiment period end up purchasing. Those purchasing spend about $75. The average user therefore spends $3.75 (95% spend $0). Assume the standard deviation is $30. If you are running an A/B test and want to detect a 5% change to revenue, you will need over 1.6 million users to achieve the desired 90% power, based on the above formula: $(4 \cdot 2 \cdot 30 \,/\, (3.75 \cdot 0.05))^2$.

If, however, you were only looking for a 5% change in conversion rate (not revenue), a lower variability OEC based on point 3.b can be used. Purchase, a conversion event, is modeled as a Bernoulli trial with p=0.05 being the probability of a purchase. The Std-Err of a Bernoulli is $\sqrt{p(1-p)}$ and thus you will need less than 500,000 users to achieve the desired power based on $\left(4 \cdot 2 \cdot \sqrt{0.05 \cdot (1-0.05)} \,/\, (0.05 \cdot 0.05)\right)^2$.

Because of the square factor, if the goal is relaxed so that you want to detect a 20% change in conversion (a factor of 4), the number of users needed drops by a factor of 16 to 30,400.

If you made a change to the checkout process, you should only analyze users who started the checkout process (point 3.c), as others could not see any difference and therefore just add noise. Assume that 10% of users initiate checkout and that 50% of those users complete it. This user segment is more homogenous and hence the OEC has lower variability. Using the same numbers as before, the average conversion rate is 0.5, the std-dev is 0.5, and thus you will need 25,600 users going through checkout to detect a 5% change based on $\left(4 \cdot 2 \cdot \sqrt{0.5 \cdot (1-0.5)} \,/\, (0.5 \cdot 0.05)\right)^2$. Since we excluded the 90% who do not initiate, the total number of users to the website should be 256,000, which is almost half the previous result, thus the experiment could run for half the time and yield the same power.

When running experiments, it is important to decide in advance on the OEC (a planned comparison); otherwise, there is an increased risk of finding what appear to be significant results by chance (familywise type I error) (6). Several adjustments have been proposed in the literature (e.g., Fisher's least-significant-difference, Bonferroni adjustment, Duncan's test, Scheffé's test, Tukey's test, and Dunnett's test), but they basically equate to

increasing the 95% confidence level and thus reducing the statistical power (15; 16; 6).

3.3 Extensions for Online Settings

Several extensions to basic controlled experiments are possible in an online setting (e.g., on the web).

3.3.1 Treatment Ramp-up

An experiment can be initiated with a small percentage of users assigned to the treatment(s), and then that percentage can be gradually increased. For example, if you plan to run an A/B test at 50%/50%, you might start with a 99.9%/0.1% split, then rampup the Treatment from 0.1% to 0.5% to 2.5% to 10% to 50%. At each step, which could run for, say, a couple of hours, you can analyze the data to make sure there are no egregious problems with the Treatment before exposing it to more users. The square factor in the power formula implies that such errors could be caught quickly on small populations and the experiment can be aborted before many users are exposed to the bad treatment.

***3.3.2** Automation*

Once an organization has a clear OEC, it can run experiments to optimize certain areas amenable to automated search. For example, the slots on the home page at Amazon are automatically optimized (3). If decisions have to be made quickly (e.g., headline optimizations for portal sites), these could be made with lower confidence levels because the cost of mistakes is lower. Multi-armed bandit algorithms and Hoeffding Races can be used for such optimizations.

3.3.3 Software Migrations

Experiments can be used to help with software migration. If a feature or a system is being migrated to a new backend, new database, or a new language, but is not expected to change user-visible features, an A/B test can be executed with the goal of retaining the Null Hypothesis, which is that the variants are not different. We have seen several such migrations, where the migration was declared complete, but an A/B test showed significant differences in key metrics, helping identify bugs in the port. Because the goal here is to retain the Null Hypothesis, it is crucial to make sure the experiment has enough statistical power to actually reject the Null Hypothesis if it false.

3.4 Limitations

Despite significant advantages that controlled experiments provide in terms of causality, they do have limitations that need to be understood. Some, which are noted in the Psychology literature are not relevant to the web (1 pp. 252-262; 17), but some limitations we encountered are certainly worth noting.

1. **Quantitative Metrics, but No Explanations.** It is possible to know which variant is better, and by how much, but not "why." In user studies, for example, behavior is often augmented with users' comments, and hence usability labs can be used to augment and complement controlled experiments (34).
2. **Short term vs. Long Term Effects.** Controlled experiments measure the effect on the OEC during the experimentation period, typically a few weeks. While some authors have criticized that focusing on a metric implies short-term focus (22) (34), we disagree. Long-term goals **should** be part of the OEC. Let us take search ads as an example. If your OEC is revenue, you might plaster ads over a page, but we know that many ads hurt the user experience, so a good OEC should include a penalty term of usage of real-estate for ads that are not clicked, and/or should directly measure repeat visits and abandonment. Likewise, it is wise to look at delayed conversion metrics, where there is a lag from the time a user is exposed to something and take action. These are sometimes called latent conversions (24; 22). Coming up with good OECs is hard, but what is the alternative? The key point here is to recognize this limitation, but avoid throwing the baby out with the bathwater.
3. **Primacy and Newness Effects**. These are opposite effects that need to be recognized. If you change the navigation on a web site, experienced users may be less efficient until they get used to the new navigation, thus giving an inherent advantage to the Control. Conversely, when a new design or feature is introduced, some users will investigate it, click everywhere, and thus introduce a "newness" bias. This bias is sometimes associated with the Hawthorne Effect (35). Both primacy and newness concerns imply that some experiments need to be run for multiple weeks. One analysis that can be done is to compute the OEC only for new users on the different variants, since they are not affected by either factor.
4. **Features Must be Implemented.** A live controlled experiment needs to expose some users to a Treatment different than the current site (Control). The feature may be a prototype that is being tested against a small portion, or may not cover all edge cases (e.g., the experiment may intentionally exclude 20% of browser types that would require significant testing). Nonetheless, the feature must be implemented and be of sufficient quality to expose users to it. Jacob Nielsen (34) correctly points out that paper prototyping can be used for qualitative feedback and quick refinements of designs in early stages. We agree and recommend that such techniques complement controlled experiments.
5. **Consistency.** Users may notice they are getting a different variant than their friends and family. It is also possible that the same user will see multiple variants when using different computers (with different cookies). It is relatively rare that users will notice the difference.
6. **Parallel Experiments.** Our experience is that strong interactions are rare in practice (33), and we believe this concern is overrated. Raising awareness of this concern is enough for experimenters to avoid tests that can interact. Pairwise statistical tests can also be done to flag such interactions automatically.
7. **Launch Events and Media Announcements**. If there is a big announcement made about a new feature, such that the feature is announced to the media, all users need to see it.

4. IMPLEMENTATION ARCHITECTURE

Implementing an experiment on a website involves two components. The first component is the *randomization algorithm*, which is a function that maps users to variants. The second component is the *assignment method*, which uses the output of the randomization algorithm to determine the experience that each user will see on the website. During the experiment, observations must be collected, and data needs to be aggregated and analyzed.

4.1 Randomization Algorithm

Finding a good *randomization algorithm* is critical because the statistics of controlled experiments assume that each variant of an experiment has a random sample of users. Specifically, the randomization algorithm should have the following four properties:

1. Users must be equally likely to see each variant of an experiment (assuming a 50-50 split). There should be no bias toward any particular variant.
2. Repeat assignments of a single user must be *consistent*; the user should be assigned to the same variant on each successive visit to the site.
3. When multiple experiments are run, there must be no correlation between experiments. A user's assignment to a variant in one experiment must have no effect on the probability of being assigned to a variant in any other experiment.
4. The algorithm should support *monotonic ramp-up* (see Section 3.3.1), meaning that the percentage of users who see a Treatment can be slowly increased without changing the assignments of users who were already previously assigned to that Treatment.

In the remainder of this section, we cover two different techniques that satisfy the above four requirements.

4.1.1 Pseudorandom with caching

A standard pseudorandom number generator can be used as the randomization algorithm when coupled with a form of caching. A good pseudorandom number generator will satisfy the first and third requirements of the randomization algorithm.

We tested several popular random number generators on their ability to satisfy the first and third requirements. We tested five simulated experiments against one million sequential user IDs, running chi-square tests to look for interactions. We found that the random number generators built into many popular languages (for example, C#) work well as long as the generator is seeded only once at server startup. Seeding the random number generator on each request may cause adjacent requests to use the same seed (as it did in our tests), which will introduce noticeable correlations between experiments. In particular, we found that the code for A/B tests suggested by Eric Peterson using Visual Basic (26) created strong two-way interactions between experiments.

To satisfy the second requirement, the algorithm must introduce state: the assignments of users must be cached once they visit the site. Caching can be accomplished either on the server side (e.g., by storing the assignments for users in some form of database), or on the client side (e.g., by storing a user's assignment in a cookie).

Both forms of this approach are difficult to scale up to a large system with a large fleet of servers. The server making the random assignment must communicate its state to all the other servers (including those used for backend algorithms) in order to keep assignments consistent.

The fourth requirement (monotonic ramp-up) is particularly difficult to implement using this method. Regardless of which approach is used to maintain state, the system would need to carefully reassign Control users who visit the site after a ramp-up to a treatment. We have not seen a system using pseudorandom-based assignment that supports ramp-up.

4.1.2 Hash and partition

Unlike the pseudorandom approach, this method is completely stateless. Each user is assigned a unique identifier, which is maintained either through a database or a cookie. This identifier is appended onto the name or id of the experiment. A hash function is applied to this combined identifier to obtain an integer which is uniformly distributed on a range of values. The range is then partitioned, with each variant represented by a partition.

This method is very sensitive to the choice of hash function. If the hash function has any *funnels* (instances where adjacent keys map to the same hash code) then the first property (uniform distribution) will be violated. And if the hash function has *characteristics* (instances where a perturbation of the key produces a predictable perturbation of the hash code), then correlations may occur between experiments. Few hash functions are sound enough to be used in this technique.

We tested this technique using several popular hash functions and a methodology similar to the one we used on the pseudorandom number generators. While any hash function will satisfy the second requirement (by definition), satisfying the first and third is more difficult. We found that only the cryptographic hash function MD5 generated no correlations between experiments. SHA256 (another cryptographic hash) came close, requiring a five-way interaction to produce a correlation. The .NET string hashing function failed to pass even a two-way interaction test.

4.2 Assignment Method

The assignment method is the piece of software that enables the experimenting website to execute a different code path for different users. A good assignment method can cause anything from visible website content to backend algorithms to vary by user. There are multiple ways to implement an assignment method. In the remainder of this section, we compare several common assignment methods and recommend best practices for their use.

4.2.1 Traffic splitting

Traffic splitting is a method that involves implementing each variant of an experiment on a different logical fleet of servers. These can be different physical servers, different virtual servers, or event different ports on the same machine. The website uses either a load balancer or proxy server to split traffic between the variants and the randomization algorithm must be embedded at this level. Traffic splitting has the advantage of being the only assignment method that requires no changes to existing code to implement an experiment. However, the approach has significant disadvantages:

1. Running experiments on small features is disproportionately difficult because the entire application must be replicated regardless of the size of the change.
2. Setting up and configuring parallel fleets is typically expensive. The Control fleet must have sufficient capacity to take 100% of the traffic in the event that the experiment needs to be shut down.
3. Running multiple experiments simultaneously is possible only with tremendous effort because the fleet must have one partition for each combination of variants across all experiments. This number increases exponentially as the number of experiments increases.
4. Any differences between the fleets used for each variant may confound the experimental results. Ideally, the hardware and network topology of each fleet will be identical and A/A tests will be run to confirm the absence of fleet-related effects. These requirements add to the already high overhead of running an experiment using traffic splitting.

The biggest drawback of traffic splitting is that it is one of the most expensive ways to implement an experiment, even though the method appears simple at first glance. We recommend this method only for testing changes that introduce significantly

different code, such as migration to a new website platform, the introduction of a new rendering engine, or a complete upgrade of a website.

4.2.2 Code-driven selection

Code-driven selection refers to a family of methods that use code embedded into the website's servers to produce a different user experience for each variant. The code takes the form of an API call placed at the point where the website logic differs between variants. The API invokes the randomization algorithm and returns the identifier of the variant to be displayed for the current user. The calling code uses this information to branch to a different code path for each variant. The API call can be placed anywhere on the server side, from front-end rendering to back-end algorithms, and a complex experiment may make use of multiple API calls inserted into the code at different places. The code changes involving the API call are specific to a single experiment and should be removed when the experiment ends. While the API can be implemented as a local function call, it typically uses an external service to ensure that the assignment logic stays consistent across a large server fleet. Code-driven assignment has four distinct advantages:

1. It is an extremely general method; it is possible to experiment on virtually any aspect of a page simply by modifying its code.
2. It places the experimentation code in the best logical place—right where decisions are made about a change. In particular, it is possible to experiment on backend features (for example, a search algorithm) without touching the front end.
3. The API can be designed to record a special event, known as a *trigger*, which indicates that a user has reached a choice point between variants, helping filter users as described in Section 3.2, point 3.c.
4. Multiple experiments can be run simultaneously without requiring any coordination between their owners.

Code-driven assignment also has a number of disadvantages, all of which are driven by its reliance on server-side code changes:

1. Because the method requires a developer to change code deep in the page logic for each experiment, implementing an experiment introduces risk. The risk is greatest on complex features whose code is spread across many pages and/or services.
2. Code changes are difficult to implement on pages that are built on content management frameworks because the API calls would need to be inserted directly into the framework and changes to frameworks are expensive. For such pages, metadata-driven selection (see section 4.2.4) is a superior approach.
3. Code changes must be manually undone to complete an experiment. While this process is trivial for a simple one-page experiment, it can be a painful process if API calls are spread throughout the code and is a big problem when experimenting on poorly designed pages.

Code-driven selection is the best approach for experiments on backend algorithms and for experiments on pages implemented directly in a language like ASP.NET. For more complex pages, a metadata-driven system (which can be built on top of code-driven selection) is superior.

4.2.3 Client-side page modification (Ajax)

Client-side page modification is the assignment technique employed by several commercial products (e.g., Google Website Optimizer). It is a method for running an experiment without making any decisions on the server. It is a form of code-driven selection in that a developer must modify code to implement an experiment. However, rather than editing code on the server side, the developer inserts JavaScript code that instructs the user's browser to invoke an assignment service at render time. The service call returns the appropriate variant for the user, and triggers a JavaScript callback that instructs the browser to dynamically alter the page being presented to the user. If done properly, the modification occurs *before* any part of the page renders. The content for each variant can either be cleverly embedded into the page or can be served by the assignment service. This method is easier to implement on a simple site than other code-driven method: all the developer needs to do is add a small snippet of JavaScript to a page. However, it has some key limitations:

1. The method does not work well for complex sites that rely on dynamic content because complex content can interact with the JavaScript code that modifies the page.
2. The use of JavaScript can add a noticeable delay for users, especially if the assignment service gets overloaded.
3. Users can determine (via the browser's View Source command) that a page is subject to experimentation, and may be able to extract the content of each variant and pollute the results.

This method is best for simple sites getting started with experimentation.

4.2.4 Metadata-driven selection

Metadata-driven selection refers to a family of techniques where the capability to experiment is built directly into a page. It can be built on top of an existing code-driven system or can be implemented as a standalone approach. The experiments themselves are configured by changing *metadata* instead of code. The metadata may be represented by anything from an editable configuration file to a relational database managed by a graphical user interface. The method is best illustrated with an example.

A content management system assembles a page from individual units called *slots*. The system refers to page metadata at render time to determine how to assemble the page. Non-technical content editors schedule pieces of content in each slot through a graphical user interface that edits this page metadata. Content can include anything from a company logo, to an image, to a snippet of text filled with links, to a widget that displays dynamic content (such as personalized recommendations). A typical experiment would be to try various pieces of content in different locations: do the recommendations receive higher click-through on the left or on the right? To enable this sort of experiment, the content management system is extended to allow pieces of content to be scheduled with respect to a specific experiment. As the page request comes in, the system executes the assignment logic for each scheduled experiment and saves the results to page context where the page assembly mechanism can react to it. The content management system only needs to be modified once; from then on, experiments can be designed, implemented, and removed by modifying the page metadata through the user interface.

In general, the metadata-driven selection method does require code changes, but it only requires them when implementing a

brand new type of experiment. Once made, these code changes never have to be undone because experiments can be activated and deactivated through the page metadata. This represents the major advantage of this method: many experiments can be implemented through metadata changes alone. The method also has drawbacks:

1. Metadata-driven selection methods limit the range of available experiments. If the content management system on a page does not support layout changes, then experiments to test out new layouts cannot be run.
2. Initial implementation is expensive because the framework must be modified to accommodate experimentation. The method is thus not suited to sites that have a large number of differently implemented pages.

Overall, this is the best method for experimentation on sites that are controlled by a single framework.

5. LESSONS LEARNED

The difference between theory and practice
is larger in practice than
the difference between theory and practice in theory
— Jan L.A. van de Snepscheut

Many theoretical techniques seem well suited for practical use and yet require significant ingenuity to apply them to messy real world environments. Controlled experiments are no exception. Having run a large number of online experiments, we now share several practical lessons in three areas: (i) analysis; (ii) trust and execution; and (iii) culture and business.

5.1 Analysis

5.1.1 Mine the Data

A controlled experiment provides more than just a single bit of information about whether the difference in OECs is statistically significant. Rich data is typically collected that can be analyzed using machine learning and data mining techniques. For example, an experiment showed no significant difference overall, but a population of users with a specific browser version was significantly worse for the Treatment. The specific Treatment feature, which involved JavaScript, was buggy for that browser version and users abandoned. Excluding the population from the analysis showed positive results, and once the bug was fixed, the feature was indeed retested and was positive.

5.1.2 Speed Matters

A Treatment might provide a worse user experience because of its performance. Greg Linden (36 p. 15) wrote that experiments at Amazon showed a 1% sales decrease for an additional 100msec, and that a specific experiments at Google, which increased the time to display search results by 500 msecs reduced revenues by 20% (based on a talk by Marissa Mayer at Web 2.0). If time is not directly part of your OEC, make sure that a new feature that is losing is not losing because it is slower.

5.1.3 Test One Factor at a Time (or Not)

Several authors (19 p. 76; 20) recommend testing one factor at a time. We believe the advice, interpreted narrowly, is too restrictive and can lead organizations to focus on small incremental improvements. Conversely, some companies are touting their fractional factorial designs and Taguchi methods, thus introducing complexity where it may not be needed. While it is clear that factorial designs allow for joint optimization of factors, and are therefore superior in theory (15; 16) our experience from running experiments in online web sites is that interactions are less frequent than people assume (33), and awareness of the issue is enough that parallel interacting experiments are avoided. Our recommendations are therefore:

- Conduct single-factor experiments for gaining insights and when you make incremental changes that could be decoupled.
- Try some bold bets and very different designs. For example, let two designers come up with two very different designs for a new feature and try them one against the other. You might then start to perturb the winning version to improve it further. For backend algorithms it is even easier to try a completely different algorithm (e.g., a new recommendation algorithm). Data mining can help isolate areas where the new algorithm is significantly better, leading to interesting insights.
- Use factorial designs when several factors are suspected to interact strongly. Limit the factors and the possible values per factor because users will be fragmented (reducing power) and because testing the combinations for launch is hard.

5.2 Trust and Execution

5.2.1 Run Continuous A/A Tests

Run A/A tests (see Section 3.1) and validate the following.

1. Are users split according to the planned percentages?
2. Is the data collected matching the system of record?
3. Are the results showing non-significant results 95% of the time?

Continuously run A/A tests in parallel with other experiments.

5.2.2 Automate Ramp-up and Abort

As discussed in Section 3.3, we recommend that experiments ramp-up in the percentages assigned to the Treatment(s). By doing near-real-time analysis, experiments can be auto-aborted if a treatment is statistically significantly underperforming relative to the Control. An auto-abort simply reduces the percentage of users assigned to a treatment to zero. By reducing the risk in exposing many users to egregious errors, the organization can make bold bets and innovate faster. Ramp-up is quite easy to do in online environments, yet hard to do in offline studies. We have seen no mention of these practical ideas in the literature, yet they are extremely useful.

5.2.3 Determine the Minimum Sample Size

Decide on the statistical power, the effect you would like to detect, and estimate the variability of the OEC through an A/A test. Based on this data you can compute the minimum sample size needed for the experiment and hence the running time for your web site. A common mistake is to run experiments that are underpowered. Consider the techniques mentioned in Section 3.2 point 3 to reduce the variability of the OEC.

5.2.4 Assign 50% of Users to Treatment

One common practice among novice experimenters is to run new variants for only a small percentage of users. The logic behind that decision is that in case of an error only few users will see a bad treatment, which is why we recommend Treatment ramp-up. In order to maximize the power of an experiment and minimize the running time, we recommend that 50% of users see each of the variants in an A/B test.

5.2.5 Beware of Day of Week Effects

Even if you have a lot of users visiting the site, implying that you could run an experiment for only hours or a day, we strongly recommend running experiments for at least a week or two, then continuing by multiples of a week so that day-of-week effects can be analyzed. For many sites the users visiting on the weekend represent different segments, and analyzing them separately may lead to interesting insights. This lesson can be generalized to other time-related events, such as holidays and seasons, and to different geographies: what works in the US may not work well in France, Germany, or Japan.

5.3 Culture and Business

5.3.1 Agree on the OEC Upfront

One of the powers of controlled experiments is that it can objectively measure the value of new features for the business. However, it best serves this purpose when the interested parties have agreed on how an experiment is to be evaluated **before** the experiment is run.

While this advice may sound obvious, it is infrequently applied because the evaluation of many online features is subject to several, often competing objectives. OECs can be combined measures, which transform multiple objectives, in the form of experimental observations, into a single metric. In formulating an OEC, an organization is forced to weigh the value of various inputs and decide their relative importance. A good technique is to assess the *lifetime value* of users and their actions. For example, a search from a new user may be worth more than an additional search from an existing user. Although a single metric is not required for running experiments, this hard up-front work can align the organization and clarify goals.

5.3.2 Beware of Launching Features that "Do Not Hurt" Users

When an experiment yields no statistically significant difference between variants, this may mean that there truly is no difference between the variants or that the experiment did not have sufficient power to detect the change. In the face of a "no significant difference" result, sometimes the decision is made to launch the change anyway "because it does not hurt anything." It is possible that the experiment is negative but underpowered.

5.3.3 Weigh the Feature Maintenance Costs

An experiment may show a statistically significant difference between variants, but choosing to launch the new variant may still be unjustified because of maintenance costs. A small increase in the OEC may not outweigh the cost of maintaining the feature.

5.3.4 Change to a Data-Driven Culture

Running a few online experiments can provide great insights into how customers are using a feature. Running frequent experiments and using experimental results as major input to company decisions and product planning can have a dramatic impact on company culture. Software organizations shipping classical software developed a culture where features are completely designed prior to implementation. In a web world, we can integrate customer feedback directly through prototypes and experimentation. If an organization has done the hard work to agree on an OEC and vetted an experimentation system, experimentation can provide real data and move the culture towards attaining shared goals rather than battle over opinions.

6. SUMMARY

Almost any question can be answered cheaply, quickly and finally, by a test campaign. And that's the way to answer them – not by arguments around a table. Go to the court of last resort – buyers of your products.
— Claude Hopkins, Scientific Advertising, 1922

Classical knowledge discovery and data mining provide insight, but the patterns discovered are correlational and therefore pose challenges in separating useful actionable patterns from those caused by "leaks" (37). Controlled experiments neutralize confounding variables by distributing them equally over all values through random assignment (6), thus establishing a causal relationship between the changes made in the different variants and the measure(s) of interest, including the Overall Evaluation Criterion (OEC). Using data mining techniques in this setting can thus provide extremely valuable insights, such as the identification of segments that benefit from a feature introduce in a controlled experiment, leading to a virtuous cycle of improvements in features and better personalization.

The basic ideas in running controlled experiments are easy to understand, but a comprehensive overview for the web was not previously available. In addition, there are important new lessons and insights that we shared throughout the paper, including generalized architectures, ramp-up and aborts, the practical problems with randomization and hashing techniques, and organizational issues, especially as they relate to OEC.

Many organizations have strong managers who have strong opinions, but lack data, so we started to use the term HiPPO, which stands for Highest Paid Person's Opinion, as a way to remind everyone that success really depends on the users' perceptions. Some authors have called experimentation the "New Imperative for Innovation" (38) and point out that "new technologies are making it easier than ever to conduct complex experiments quickly and cheaply." We agree and believe that companies can accelerate innovation through experimentation because it is the customers' experience that ultimately matters.

ACKNOWLEDGMENTS

We would like to thank members of the Experimentation Platform team at Microsoft. The first author wishes to thank Avinash Kaushik for a great conversation on A/B testing, where he used the term "HiPO" for Highest Paid Opinion; this evolved into HiPPO (picture included) in our presentations. We thank Fritz Behr, Keith Butler, Rob Deyo, Danyel Fisher, James Hamilton, Ronit HaNegby, Greg Linden, Foster Provost, Saharon Rosset, and Zijian Zheng for their feedback.

7. REFERENCES

1. **Rossi, Peter H, Lipsey, Mark W and Freeman, Howard E.** *Evaluation: A Systematic Approach.* 7th. s.l. : Sage Publications, Inc, 2003. 0-7619-0894-3.

2. **Linden, Greg.** Early Amazon: Shopping cart recommendations. *Geeking with Greg.* [Online] April 25, 2006. http://glinden.blogspot.com/2006/04/early-amazon-shopping-cart.html.

3. **Kohavi, Ron and Round, Matt.** *Front Line Internet Analytics at Amazon.com.* [ed.] Jim Sterne. Santa Barbara, CA : s.n., 2004. http://ai.stanford.edu/~ronnyk/emetricsAmazon.pdf.

4. **Thomke, Stefan H.** *Experimentation Matters: Unlocking the Potential of New Technologies for Innovation.* s.l. : Harvard Business School Press, 2003. 1578517508.

5. **Charles, Reichardt S and Melvin, Mark M.** Quasi Experimentation. [book auth.] Joseph S Wholey, Harry P Hatry and Kathryn E Newcomer. *Handbook of Practical Program Evaluation.* 2nd. s.l. : Jossey-Bass, 2004.

6. **Keppel, Geoffrey, Saufley, William H and Tokunaga, Howard.** *Introduction to Design and Analysis.* 2nd. s.l. : W.H. Freeman and Company, 1992.

7. **Eisenberg, Bryan.** How to Decrease Sales by 90 Percent. *ClickZ.* [Online] Feb 21, 2003. http://www.clickz.com/showPage.html?page=1588161.

8. —. How to Increase Conversion Rate 1,000 Percent. *ClickZ.* [Online] Feb 28, 2003. http://www.clickz.com/showPage.html?page=1756031.

9. **McGlaughlin, Flint, et al.** The Power of Small Changes Tested . *Marketing Experiments Journal.* [Online] March 21, 2006. http://www.marketingexperiments.com/improving-website-conversion/power-small-change.html.

10. **Usborne, Nick.** Design Choices Can Cripple a Website. *A List Apart.* [Online] Nov 8, 2005. http://alistapart.com/articles/designcancripple.

11. **Eisenberg, Bryan and Eisenberg, Jeffrey.** *Call to Action, Secret formulas to improve online results.* Austin, Texas : Wizard Academy Press, 2005. Making the Dial Move by Testing, Introducing A/B Testing.

12. **Eisenberg, Bryan; Garcia, Anthony;.** Which Sells Best: A Quick Start Guide to Testing for Retailers. *Future Now's Publications.* [Online] 2006. http://futurenowinc.com/shop/.

13. **Chatham, Bob, Temkin, Bruce D and Amato, Michelle.** *A Primer on A/B Testing.* s.l. : Forrester Research, 2004.

14. **Forrester Research.** *The State of Retailing Online.* s.l. : Shop.org, 2005.

15. **Mason, Robert L, Gunst, Richard F and Hess, James L.** *Statistical Design and Analysis of Experiments With Applications to Engineering and Science.* s.l. : John Wiley & Sons, 1989. 047185364X .

16. **Box, George E.P., Hunter, J Stuart and Hunter, William G.** *Statistics for Experimenters: Design, Innovation, and Discovery.* 2nd. s.l. : John Wiley & Sons, Inc, 2005. 0471718130.

17. **Weiss, Carol H.** *Evaluation: Methods for Studying Programs and Policies.* 2nd. s.l. : Prentice Hall, 1997. 0-13-309725-0.

18. **Roy, Ranjit K.** *Design of Experiments using the Taguchi Approach : 16 Steps to Product and Process Improvement.* s.l. : John Wiley & Sons, Inc, 2001. 0-471-36101-1.

19. **Peterson, Eric T.** *Web Analytics Demystified: A Marketer's Guide to Understanding How Your Web Site Affects Your Business.* s.l. : Celilo Group Media and CafePress, 2004. 0974358428.

20. **Eisenberg, Bryan.** How to Improve A/B Testing. *ClickZ Network.* [Online] April 29, 2005. http://www.clickz.com/showPage.html?page=3500811.

21. —. A/B Testing for the Mathematically Disinclined. *ClickZ.* [Online] May 7, 2004. http://www.clickz.com/showPage.html?page=3349901.

22. **Quarto-vonTivadar, John.** AB Testing: Too Little, Too Soon. *Future Now.* [Online] 2006. http://www.futurenowinc.com/abtesting.pdf.

23. **Miller, Scott.** How to Design a Split Test. *Web Marketing Today, Conversion/Testing.* [Online] Jan 18, 2007. http://www.wilsonweb.com/conversion/.

24. —. *The ConversionLab.com: How to Experiment Your Way to Increased Web Sales Using Split Testing and Taguchi Optimization.* 2006. http://www.conversionlab.com/.

25. **Kaushik, Avinash.** Experimentation and Testing: A Primer. *Occam's Razor by Avinash Kaushik.* [Online] May 22, 2006. http://www.kaushik.net/avinash/2006/05/experimentation-and-testing-a-primer.html.

26. **Peterson, Eric T.** *Web Site Measurement Hacks.* s.l. : O'Reilly Media, 2005. 0596009887.

27. **Tyler, Mary E and Ledford, Jerri.** *Google Analytics.* s.l. : Wiley Publishing, Inc, 2006. 0470053852.

28. **Sterne, Jim.** *Web Metrics: Proven Methods for Measuring Web Site Success.* s.l. : John Wiley & Sons, Inc, 2002. 0-471-22072-8.

29. **Kaplan, Robert S and Norton, David P.** *The Balanced Scorecard: Translating Strategy into Action.* s.l. : Harvard Business School Press, 1996. 0875846513.

30. **Ulwick, Anthony.** *What Customers Want: Using Outcome-Driven Innovation to Create Breakthrough Products and Services.* s.l. : McGraw-Hill, 2005. 0071408673.

31. *Portable Power.* **Wheeler, Robert E.** 1974, Technometrics, Vol. 16. http://www.bobwheeler.com/stat/Papers/PortablePower.PDF.

32. *The Validity of Portable Power.* **Wheeler, Robert E.** 2, May 1975, Technometrics, Vol. 17, pp. 177-179.

33. **van Belle, Gerald.** *Statistical Rules of Thumb.* s.l. : Wiley-Interscience, 2002. 0471402273.

34. **Nielsen, Jakob.** Putting A/B Testing in Its Place. *Useit.com Alertbox.* [Online] Aug 15, 2005. http://www.useit.com/alertbox/20050815.html.

35. Hawthorne effect. *Wikipedia.* [Online] 2007. http://en.wikipedia.org/wiki/Hawthorne_experiments.

36. **Linden, Greg.** Make Data Useful. [Online] Dec 2006. http://home.blarg.net/~glinden/StanfordDataMining.2006-11-29.ppt.

37. *Lessons and Challenges from Mining Retail E-Commerce Data.* **Kohavi, Ron, et al.** 1-2, s.l. : Kluwer Academic Publishers, 2004, Machine Learning, Vol. 57, pp. 83-113. http://ai.stanford.edu/~ronnyk/lessonsInDM.pdf.

38. *Enlightened Experimentation: The New Imperative for Innovation.* **Thomke, Stefan.** Feb 2001, Harvard Business Review . R0102D.