OPENPRISE
Data Automation for Everybody

We've all heard these sayings applied to business, like "KISS: Keep it simple, stupid," and "the simple solution's usually the best solution." There's a lot of truth to each of them. Take "simple doesn't mean easy." It's also true, but it can be difficult to understand why people often confuse "simple" with "simplistic." In sales ops and marketing ops (let's use RevOps as a shorthand for the rest of this article), people often mistake a simplistic RevOps solution for a simple solution—when it's usually the reverse. Let's examine the differences, illustrated with a few common RevOps examples.

**Defining Simple vs. Simplistic**

**A SIMPLE** RevOps solution is one thats simple for both the end user and the operator. A simple solution looks like this:

- Easy to comprehend in concept
- Not controversial, most users agree it's "the right thing to do"
- Easy to execute without much instruction, training, or enforcement
- Users can explain to others without coaching
- Users want to use it and the solution endures the test of time

**A SIMPLISTIC** RevOps solution is one that's quick and easy to implement for the implementor, but the resulting solution often looks like this:

- Users frequently question the usefulness of the solution or are confused about how to use it effectively
- Controversial based on the way different users' function and their motivation
- Requires significant documentation, training, and enforcement to work
- Users either can't explain to others without coaching or don't want to because they're skeptical of it or don't really understand it
- User don't accept it and don't want to use it, but are told to use it, and often try to work around it

**Simple is Usually Not Easy**

In RevOps and many other situations in business, a simple solution is usually not easy to design and implement. So, a simple solution is hardly a simplistic solution. Worse yet, solution vendors too often try to sell simplistic solutions as the simple solution, and too many RevOps buyers—especially

inexperienced buyers who haven't earned their battle scars—are none the wiser to the difference between a simple solution and a simplistic solution.

A simple solution, once implemented, can be a beautiful thing. Its adoption will grow virally, and it will stand the test of time even within the most complex organizations. However, designing and implementing a simple solution can be anything but easy and simplistic. In RevOps especially, the challenge to having a simple solution is often data—either the lack of data, or the lack of "useful" data.

Let's illustrate this with a few common examples.

#### Job Segmentation vs. Job Title Standardization

Job title is a frustrating piece of data. It's such a rich piece of data, as it tells you a lot about who the prospect is, what they do, and whether they're the right person for your sales team to pursue or for your marketing team to target. Yet, it's a very difficult piece of data to work with because it:

- Is unstructured data
- Uses different and inconsistent abbreviations: VP, Sr. Dir, CEO, C.E.O.
- Lacks context: Assistant to the VP, Engineer in the CTO Office
- Tries too hard to be creative: what exactly is a Growth Hacker or a Revenue Consultant?
- Can contain industry-specific context: a Director is more senior than a VP in financial services

So, we can't use raw job title data for any analytics or automation needs. The first impulse for many RevOps folks might be to standardize on job title, in other words, transform all these variations into Vice President of Human Resources

- Vice President of HR
- V.P. Human Resources
- VP, People, North America

Then they compile a list of every variation of these job titles they've ever seen to do a search and replace. This is a **simplistic** RevOps solution because the

design and implementation is quick and simple. But it's difficult to manage and scale, and the end result is limited in usefulness.

A **simple** RevOps solution, from the end user's standpoint, is not to deal with job title at all, but instead use structured and abstracted job function, job sub-function, job level, and buyer persona. So instead of "Vice President of Human Resources, Pension and Retirement, Americas," the much more usable data is:

- Job level = Executive
- Job function = Human Resources
- Job sub-function = Benefits
- Buyer personal = Budget Owner

This data is ready to use for analysis, modeling, and automation. It's a simple solution. But it takes a lot of technology, job keyword data, and know-how to turn any old job title into structured data reliably, so it's not an easy solution and definitely not a simplistic solution.

#### Incremental Scoring vs. Surge Scoring

If your organization does any scoring at all, it's probably incremental behavior scoring, looks something like this:

- Prospect opens a marketing email = 2 points
- Prospect downloads a white paper = 10 points
- Prospect has a conversation at conference = 30 points
- Prospect fills out contact form = 100 points
- Prospect unsubscribes = -100 points

This **simplistic** RevOps solution can be implemented quickly and easily with any marketing automation platform like Marketo and Eloqua. It's simplistic because it's unclear what this score actually means and how it would be useful to a salesperson. Take these three prospects with exactly the same score of 100:

1. A marketing student who downloads a white paper offered in your newsletter every month, but won't be buying your

product any time soon.
2. A marketing director from a company that fits your ideal customer profile who's never engaged before, watched two videos and downloaded a buyer's guide within a span of three days, then goes quiet again.
3. A prospect who's really engaged for two months, is in the process of evaluating your product, unsubscribes to your marketing email.

In most companies, sales will likely tell you they don't think marketing's scoring is very useful and they're not sure how to use the score effectively to improve their ability to prospect.

In contrast, a scoring model that can evaluate a broad range of engagement and assess every engagement's value based not just on what the engagement is, but also how old the engagement is and what the engagement pattern looks like, is a **simple** and useful RevOps solution for the sales team. Building off the examples above, a surge model based on sales and marketing engagement can provide more useful insights:

1. A prospect with a steady, long-running level of engagement shows strong interest, but no sign of "in-the-market" to purchase.
2. A prospect that shows a surge of activity concentrated in time and significantly above their baseline engagement level is potentially an "in-the-market" prospect who's researching for a high-priority project
3. A prospect that's actively engaged on the sales side but decided to stop receiving marketing communications may be an indication that they've made the purchasing decision and decided there'is no need to consume any more marketing assets.

A useful surge model can tell the sales team that Prospect #1 is nothing to get excited about, Prospect #2 is worth chasing and time-sensitive, and Prospect #3 is still highly engaged but no need to sound the alarm. While it's more difficult to implement this type of scoring engine, it's simple for the sales team because a high score corresponds to a better lead to them, without their having to dig into the detailed engagement history to really understand whether a lead is really good despite the high score.

#### Last-Touch Attribution … for What?

Among all the analysis a RevOps team can do, attribution is probably the most important. Attribution is a feedback mechanism that tells you what works and what doesn't, so you can invest more in what works and attempt to fix or

improve on what doesn't. People generally analyze three types of attribution: first touch, multi-touch, and last touch.

First-touch: the marketing activity that added a new prospect into the database. This is clear and not very controversial.

Multi-touch: measures the nurturing engagements that contributed to a successful sale. While RevOps professionals love to debate the merits of different multi-touch distribution models, it's largely a religious debate. None are perfect and no one can really prove the superiority of one model over another. Any measurement's better than no measurement in this case.

Last-touch the last marketing activity that preceded the creation of an opportunity in the CRM. Sounds logical enough. Last-touch can be useful for low-price B2B sales, but makes absolutely no sense in higher priced B2B sales where the salesperson is more than just an order taker.

In this type of sale, the salesperson always creates the opportunity following some type of qualification conversation. For Marketing to claim that any marketing activity triggers converting a suspect to a prospect is at best self-delusional. We can fairly confidently state that the activity that should receive last-touch attribution is always a sales activity. While that's accurate, it's hardly insightful and useful data.

Instead, **simple** last-touch attribution is the last activity that turned a one-way engagement into a two-way engagement, for example:

- A webinar that caused the prospect to contact sales
- The fifth prospecting email that finally elicited a positive response from the prospect
- An ad click 5 minutes before the prospect decided to pick up on the fourth cold-call

This type of last-touch attribution is simple to understand and offers demonstrably valuable insight. But gathering all the necessary data to enable this type of attribution and ensure accurate results is very difficult and very few companies have been able to achieve it.

**Knowing the Difference Between Simple and Simplistic RevOps Solutions**

The unfortunate truth is that simple RevOps solutions are hardly ever easy. It takes time and effort to think through the design, get all the data needed to support the solution, and implement the process and technology to make it sustainable.

So why is it important? While you may conclude that a simple solution is too difficult to do (and the organization doesn't have the appetite to do it at the moment), it's still valuable to really understand the difference between a meaningless simplistic solution vs. a useful, simple, but hard-to-implement solution. Spending time and resources to build a simplistic solution is not only a waste of time and valuable resources, but can erode trust and confidence between teams. And in the worst case, it provides the wrong feedback to leadership about continuing to invest in stuff that doesn't work or could potentially harm the business.