The Evolution of Continuous Experimentation in Software Product Development

From Data to a Data-driven Organization at Scale

Aleksander Fabijan
Pavel Dmitriev
Helena Holmström Olsson
Jan Bosch
The Evolution of Continuous Experimentation in Software Product Development

From **Data** to a **Data-driven** Organization at **Scale**

Aleksander Fabijan
Pavel Dmitriev
Helena Holmström Olsson
Jan Bosch
How to evolve *controlled experimentation* to become data-driven at scale?
Key Learnings

1. The journey from a company with data to a data-driven company at scale is an evolution.

2. The experimentation does not need to be complex (at first)!

3. Trustworthiness enables scalability, and not the other way around!
Agenda

• Background & Motivation
• Research Method
• The Experimentation Evolution Model
  • Crawl stage
  • Walk stage
  • Run stage
  • Fly stage
• Conclusions and Q&A
Background & Motivation

• Software companies are increasingly aiming to become data-driven.

• Getting data is easy. Getting data that you can trust? Not that much...

• What customers say they want/do differs from what they actually want/do.

• Online connectivity of products opens new opportunities to collect and use data that reveals what the customers actually want/do,

• Online Controlled Experiments (e.g. A/B tests) can enable Software Companies to more accurately identify what delivers value to their customers.
Twist...

**Problem:** Experimentation in large software companies *is challenging.*
- Running a few A/B tests is *simple.* Scaling Experimentation *not so much!*
- Challenges include *instrumentation,* data loss, data pipelines, assumption violations of classical statistical methods, finding the right *metrics,* etc.
Problem/Solution

• **Problem**: Experimentation in large software companies is *challenging*.
  • Running a few A/B tests is *simple*. Scaling Experimentation *not so much*!
  • Challenges include *instrumentation*, data loss, data pipelines, assumption violations of classical statistical methods, finding the right *metrics*, etc.

• **Solution**: We provide *step-by-step guidance* on how to *develop* and *evolve* the experimentation practices (*technical, organizational and business*).
Research Method

• **Inductive case study** conducted in collaboration with the Analysis and Experimentation team.

• **Data Collection:**
  • The study is based on historical data points *(past experiments)*, and
  • Complemented with a series of semi-structured interviews, observations, and meeting participations.

• **Data Analysis:**
  • We grouped the collected data in four buckets, and performed iterative category development to emerge with the three levels of evolution.
The Experimentation Evolution Model

• Our model provides guidance on how to become data-driven at scale through the evolution of online controlled experimentation.

• We identify four stages of experimentation evolution:

  - Crawl
  - Walk
  - Run
  - Fly

• We identify the most important R&D activities to focus on in each of the stages in order to advance.
**The Experimentation Evolution Model**

<table>
<thead>
<tr>
<th>Category/Phase</th>
<th>Crawl</th>
<th>Walk</th>
<th>Run</th>
<th>Fly</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Technical focus of product dev. Activities</strong></td>
<td>(1) Logging of signals</td>
<td>(1) Setting-up a reliable pipeline</td>
<td>(1) Learning experiments</td>
<td>(2) Standardized process for metric design and evaluation, and OEC improvement</td>
</tr>
<tr>
<td>(2) Work on data quality issues</td>
<td>(2) Creation of simple metrics</td>
<td>(2) Comprehensive metrics</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3) Manual analysis of experiments</td>
<td>Combining signals with analysis units. Four types of metrics are created: debug metrics (largest group), success metrics, guardrail metrics and data quality metrics.</td>
<td>Creation of comprehensive set of metrics using the knowledge from the learning experiments.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Experimentation platform complexity</strong></td>
<td>No experimentation platform</td>
<td>Platform is required</td>
<td>New platform features</td>
<td>Advanced platform features</td>
</tr>
<tr>
<td>An initial experiment can be coded manually (adhoc).</td>
<td>3rd party platform can be used or internally developed. The following two features are required:</td>
<td>The experimentation platform should be extended with the following features:</td>
<td>The following features are needed:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Power Analysis</td>
<td>• Alerting</td>
<td>• Interaction control and detection</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Pre-Experiment A/A testing</td>
<td>• Control of carry-over effect</td>
<td>• Near real-time detection and automatic shutdown of harmful experiments</td>
<td></td>
</tr>
<tr>
<td><strong>Experimentation pervasiveness</strong></td>
<td>Generating management support</td>
<td>Experiment on individual feature level</td>
<td>Expanding to (1) more features and (2) other products</td>
<td>Experiment with every minor change to portfolio</td>
</tr>
<tr>
<td>Experimenting with e.g. design options for which it’s not a priori clear which one is better. To generate management support to move to the next stage.</td>
<td>Broadening the types of experiments run on a limited set of features (design to performance, from performance to infrastructure experiments)</td>
<td>Experiment on most new features and most products.</td>
<td>Experiment with any change on all products in the portfolio. Even to e.g. small bug fixes on feature level.</td>
<td></td>
</tr>
<tr>
<td><strong>Engineering team self-sufficiency</strong></td>
<td>Limited understanding</td>
<td>Creation and set-up of experiments</td>
<td>Creation and execution of experiments</td>
<td>Creation, execution and analysis of experiments</td>
</tr>
<tr>
<td>External Data Scientist knowledge is needed in order to set-up, execute and analyse a controlled experiment.</td>
<td>Creating the experiment (instrumentation, A/A testing, assigning traffic) is managed by the local Experiment Owners. Data scientists responsible for the platform supervise Experiment Owners and correct errors.</td>
<td>Includes monitoring for bad experiments, making ramp-up and shut-down decisions, designing and deploying experiment-specific metrics.</td>
<td>Scorecards showing the experiment results are intuitive for interpretation and conclusion making.</td>
<td></td>
</tr>
<tr>
<td><strong>Experimentation team organization</strong></td>
<td>Standalone</td>
<td>Data science team that implemented the platform supports different product teams and their Experiment Owners. Product teams do not have their own data scientists that would analyse experiments independently.</td>
<td>Product teams hire their own data scientists that create a strong unity with business. Learning between the teams is limited to their communication.</td>
<td>Small data science teams in each of the product teams.</td>
</tr>
<tr>
<td>Fully centralized data science team. In product teams, however, no or very little data science skills. The standalone team needs to train the local product teams on experimentation. We introduce the role of Experiment Owner (EO).</td>
<td></td>
<td></td>
<td>Learnings from experiments are shared automatically across organization via the Institutional memory features.</td>
<td></td>
</tr>
<tr>
<td><strong>Business Evaluation Criteria (OEC)</strong></td>
<td>OEC is defined for the first set of experiments with a few key signals that will help ground expectations and evaluation of the experiment results.</td>
<td>OEC evolves from a few key signals to a structured set of metrics consisting of Success, Guardrail and Data Quality metrics. Debug metrics are not a part of OEC.</td>
<td>OEC is tailored with the findings from the learning experiments. Single metric as a weighted combination of others is desired.</td>
<td>OEC is stable, only periodic changes allowed (e.g. 1 per year). It is also used for setting the performance goals for teams within the organization.</td>
</tr>
</tbody>
</table>
# The Experimentation Evolution Model

<table>
<thead>
<tr>
<th>Category/Phase</th>
<th>Crawl</th>
<th>Walk</th>
<th>Run</th>
<th>Fly</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technical focus of product dev. Activities</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- (1) Logging of signals that will help ground expectations and evaluation of the experiment results. 
- (2) Work on data quality issues 
- (3) Manual analysis of experiments 
Transitioning from the debugging logs to a format that can be used for data-driven development. | 
- (1) Setting-up a reliable pipeline 
- (2) Creation of simple metrics 
Combining signals with analysis units. Four types of metrics are created: debug metrics (largest group), success metrics, guardian metrics, and data quality metrics. | 
- (1) Learning experiments 
- (2) Comprehensive metrics 
Creation of a comprehensive set of metrics using the knowledge from the learning experiments. | 
- (1) Standardized process for metric design and evaluation, and OEC improvement |
| Experimentation platform complexity | 
No experimentation platform 
An initial experiment can be coded manually (ad-hoc). | 
Platform is required 
3rd party platform can be used or internally developed. The following two features are required: 
- Power Analysis 
- Pre-Experiment A/A testing | 
New platform features 
The experimentation platform should be extended with the following features: 
- Alerting 
- Control of carry-over effect 
- Experiment iteration support | 
Advanced platform features 
The following features are needed: 
- Interaction control and detection 
- Near real-time detection and automatic shutdown of harmful experiments 
- Institutional memory |
| Experimentation pervasiveness | 
Generating management support 
Experimenting with e.g. design options for which it’s not a priori clear which one is better. To generate management support to move to the next stage. | 
Experiment on individual feature level 
Broadening the types of experiments run on a limited set of features (design to performance, from performance to infrastructure experiments) | 
Expanding to (1) more features and (2) other products 
Experiment on most new features and most products. | 
Experiment with every minor change to portfolio 
Experiment with any change on all products in the portfolio. Even to e.g. small bug fixes on feature level. |
| Engineering team self-sufficiency | 
Limited understanding 
External Data Scientist knowledge is needed in order to set up, execute and analyse a controlled experiment. | 
Creation and set-up of experiments 
Creating the experiment (instrumentation, A/A testing, assigning traffic) is managed by the local Experiment Owners. Data scientists responsible for the platform supervise Experiment Owners and correct errors. | 
Creation and execution of experiments 
Includes monitoring for bad experiments, making ramp-up and shut-down decisions, designing and deploying experiment-specific metrics. | 
Creation, execution and analysis of experiments 
Scorecards showing the experiment results are intuitive for interpretation and conclusion making. |
| Experimentation team organization | 
Standalone 
Fully centralized data science team. In product teams, however, no or very little data science skills. The standalone team needs to train the local product teams on experimentation. We introduce the role of Experiment Owner (EO). | 
Data science team that implemented the platform supports different product teams and their Experiment Owners. Product teams do not have their own data scientists that would analyse experiments independently. | 
Product teams hire their own data scientists that create a strong unity with business. Learning between the teams is limited to their communication. | 
Small data science teams in each of the product teams. 
Learnings from experiments are shared automatically across organization via the institutional memory features. |
| Business Evolution Criteria (OEC) | 
OEC is defined for the first set of experiments with a few key signals that will help ground expectations and evaluation of the experiment results. | 
OEC evolves from a few key signals to a structured set of metrics consisting of Success, Guardian and Data Quality metrics. Debug metrics are not a part of OEC. | 
OEC is tailored with the findings from the learning experiments. Single metric as a weighted combination of others is desired. | 
OEC is stable, only periodic changes allowed (e.g. 1 per year). It is also used for setting the performance goals for teams within the organization. |
Crawl stage (1x experiments yearly)

• Technical focus:
  • **Logging** of signals (clicks, dwell times, swipes, etc.) should be implemented,
  • **Trustworthiness** of collected data should be considered (data quality),
  • Analysis of the experiment results can be done **manually**.

• Team Focus:
  • Product teams gain **management support** with the first experiments.

• Business focus:
  • The *Overall Evaluation Criteria* consists of **a few key signals**.
Contextual Bar Experiment

• **Experiment Goal:**
  • Identify whether the *contextual command bar* improves editing efficiency.

• **Value Hypotheses:**
  • (1) increased commonality and frequency of edits,
  • (2) increased 2-week retention.

• **Outcome:**
  • The initial experiment was unsuccessful due to **logging misconfiguration**.
Walk stage (10x experiments yearly)

• Technical focus:
  • Starting to develop/integrate an experimentation platform.
  • Defining **success** metrics, **debug** metrics, **guardrail** metrics, and **data** quality metrics,

• Team focus:
  • Product team **designs and executes** experiments related to their features.

• Business focus:
  • The Overall Evaluation Criteria in this stage evolves from **signals** to a structured set of **metrics** (guardrail, success, and data-quality metrics)
The “Xbox deals” experiment

• **Experiment Goal:**
  - Identify the impact of showing the discount in the weekly deals stripe.

• **Value Hypotheses:**
  - (1) increased engagement with the stripe
  - (2) no decrease in purchases.

• **Outcome:**
  - Treatment C increased both engagement with the stripe and purchases made.
Run stage (100x experiments yearly)

• Technical focus:
  • Features: alerting, control of carry over effects, experiment iteration, etc.
  • Learning experiments: Create comprehensive metrics.

• Team Focus:
  • Experimentation expands to other feature teams (and other products),
  • Teams create, execute and monitor experiments.

• Business focus:
  • The Overall Evaluation Criteria is tailored using the learning experiments.
The “MSN.com personalization” experiment

• **Experiment Goal:**
  • Identify the impact of ML sorting of articles in comparison to editor curated articles.

• **Value Hypotheses:**
  • (1) ML curated articles increase engagement

• **Outcome:**
  • At first, ML articles performed worse than editor curating. After a few iterations things changed!
Fly stage (1000+ experiments yearly)

• Technical focus:
  • **Features**: interaction between experiments, autonomous shutdown, and an accumulation of institutional memory.
  • Experiment results should become **intuitive** (e.g. green for go / red for no-go)

• Team Focus:
  • Product Teams experiment with every minor change in the portfolio, even the **smallest bug fixes**.

• Business focus:
  • The Overall Evaluation Criteria is **stable**.
  • OEC can become used to set the **performance goals for teams** and a measure of their success.
Bing Bot Detection Experiment

• **Experiment Goal:**
  • Evaluate the improved detection of bots.

• **Value Hypotheses:**
  • (1) No change to real user experience,
  • (2) Fewer resources used to compute search results.

• **Outcome:**
  • ~10% saving on infrastructure resources.
Wicked Skengman @Russtafarii · 3 Sep 2015

this is why people don't use bing @Microsoft @bing @BingAds
Conclusions

1. The **journey** from a company with data to a data-driven company at scale is an **evolution**:
   - (1) *culture*, (2) *business*, and the (3) *technical capabilities*.

2. The experimentation starts ‘easy’ and becomes challenging!
   - The need for a more *detailed training* arises **when many experiments are being executed by many** product teams
   - The need for a more sophisticated platform arises when many teams experiment and **interfere with each other**

1. Trustworthiness enables scalability, and **not** the other way around!
Thank you!