Create a basic prioritization framework

Skip Ahead

1. Define your prioritization criteria
2. Prioritize your list
3. Review your process

Automate idea submissions and ranking

This article will help you:

- Identify the right prioritization framework for your testing team and download a template
- Organize your list of test ideas from first-to-run to last

How do you decide what to optimize and when? To figure out which experiments and campaigns to run first and which to place into your backlog, use a prioritization framework to evaluate your ideas.

A basic prioritization framework uses consistent criteria to order the experiments and campaigns you'll run, from first to last. You'll use this framework to manage your backlog and experiment cycles. Use a prioritization framework to ensure that your most impactful tests run first.

- This basic template for prioritizing ideas (Excel) can help you get started. A concept called minimum detectable effect (MDE) can also help you prioritize tests based on expected ROI.
- A full roadmap includes a scoring rubric and an execution timeline.

Read on to learn about basic prioritization, as well as how to automate your process.

Optimizely's Program Management feature, available on select plans, enables you to scale an experimentation program across an enterprise and gain program-level reporting.

See requirements

Materials to prepare
• List of test ideas including:
  ▪ Dependencies
  ▪ Effort estimates
  ▪ Impact estimates
  ▪ Time to significance and available traffic
  ▪ Likelihood of implementation
  ◦ Criteria for prioritization
  ◦ Resourcing constraints
  ◦ Insights from technical audit

People and resources

  • Program Manager (responsible for final scoring)
  ◦ Developer (responsible for estimating effort)
  ◦ Designer (responsible for estimating effort)
  ◦ Executive sponsor (review, approve, and provide strategic alignment)

Actions you’ll perform

  • Score impact versus effort
    ◦ Use MDE to estimate impact versus effort
  ◦ Add experiment tags or category labels
  ◦ Rank experiments
  ◦ Create a balanced approach that optimizes for different goals
  ◦ Schedule tests and campaigns in the roadmap
  ◦ Document the prioritization process
  ◦ Socialize an optimization culture at the company

Deliverables

  • A prioritized list of experiments and campaigns, or an advanced roadmap

What to watch out for

  • It can be difficult to quantify the impact of experiments
    ◦ If you don't use a prioritization scheme, you may end up prioritizing ideas according to dominant trends in the company, top-down
    ◦ Without outlining dependencies in advance, you may slow down testing or be unable to run a test at all
    ◦ A lack of documentation can slow down a team with a large roadmap
    ◦ An unbalanced roadmap can over-index certain parts of the site and leave other opportunities on the table -- balance by UX theme, location, tactics, and goals pursued
    ◦ Roadmaps that are entirely agile or entirely waterfall each present difficulties in planning
This article is part of the Optimization Methodology series.

If you're unsure of the extent of your organization's experimentation capabilities, Optimizely's Maturity Model can help you determine what kind of experiments you should be doing, as well as the next steps your company should take to maximize your returns from experimentation.

1. Define your prioritization criteria

We suggest you evaluate ideas based on two factors: impact and effort. What counts as high impact, or low effort? This depends on your company's business goals and your team's access resources.

Impact: What metrics will you use to measure the success of your optimization program? Which events in Optimizely directly influence these metrics?

<table>
<thead>
<tr>
<th>Impact Factors</th>
<th>Soft Impact (Meaningful)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Higher conversion rate: Purchases, Sign-ups, Engagement</td>
<td>Internal buy-in, Executive sponsorship</td>
</tr>
<tr>
<td>Metrics you want to maximize: Revenue, Pageviews</td>
<td>Experiment (test ideas from different areas of your company), Resource allocation (dedicated developer)</td>
</tr>
<tr>
<td>Metrics you want to reduce: Internal costs</td>
<td>Access (places where testing can or should happen), Implementation</td>
</tr>
</tbody>
</table>

Effort: What is easy or difficult to do? Which resources are dedicated to testing and which are shared or borrowed from other teams?

<table>
<thead>
<tr>
<th>Effort Factors</th>
<th>Teams (or People)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technical</td>
<td>Power users, Designers, Developers, Graphic designers, Photography team, SEO and/or SEM team, IT department, QA department, Executive sponsorship</td>
</tr>
<tr>
<td>Teams or People</td>
<td>HTML, CSS, JavaScript, Backend changes (reordering pages in a checkout flow), Advanced audience targeting conditions (cookies, ad campaigns, custom tags, etc.) Time to run</td>
</tr>
</tbody>
</table>

Team members to consult:
• The program manager, who is responsible for the overall framework and final scoring
• The developer, who is responsible for estimating effort
• The design team, who is responsible for estimating effort
• The executive sponsor, who will review and approve prioritized list, and provide strategic alignment

Ultimately, the criteria you use to prioritize your ideas will depend on your particular program’s goals and resources.

For example, your team may be technically savvy but low on design resources, so you set up tests easily but have trouble getting mockups. Or if you have executive buy-in but find it difficult to get time with your developers, you may find that advanced test ideas are quickly greenlit but slow to be implemented.

These types of factors are important to consider when deciding on the criteria for effort and impact.

2. Prioritize your list

Assign effort scores and impact scores to every optimization idea and prioritize accordingly.

You can use broad categories like high, medium, and low when evaluating impact and effort.

**High-impact, low-effort** tests and campaigns should run first.

![Prioritization by Effort vs. Impact](image)

Or, you can assign numerical scores; scores can help provide a more granular view of the relative ROI of each experiment. This is known as a blended model.

Assign numerical values for effort and impact. Then, simply sum the impact and effort scores for each test and campaign to generate a single prioritization score that combines both sets of criteria.
Note:

In the example above, **high impact** tests and campaigns are given **high numerical scores**. However, **high effort** ideas are given **low numerical scores**.

When we sum the two scores, high-impact, low-effort ideas rise to the top of the prioritized list. These tests and campaigns should run first.

You can further enforce consistency in your prioritization process by building out a detailed rubric. Customize the weights of your effort scores to the strengths of your team. Adjust the weights of your impact scores according to the goals that are most important to your business.

<table>
<thead>
<tr>
<th>Sample Blended Prioritization Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Priority</strong></td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

Score = Effort + Impact

<table>
<thead>
<tr>
<th>Sample Impact Rubric</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Impact Component</strong></td>
</tr>
<tr>
<td>high impact</td>
</tr>
<tr>
<td>Conversion rate</td>
</tr>
<tr>
<td>Excitement</td>
</tr>
<tr>
<td>Implementation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sample Effort Rubric</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Effort Component</strong></td>
</tr>
<tr>
<td>low effort</td>
</tr>
<tr>
<td>HTML</td>
</tr>
<tr>
<td>Javascript / JQuery</td>
</tr>
<tr>
<td>Graphic Design</td>
</tr>
<tr>
<td>Executive Sponsor</td>
</tr>
</tbody>
</table>

With a rubric, you can consistently and objectively prioritize all your test and campaign.
Tip:

At this stage, you might also evaluate additional attributes beyond effort and impact. To learn more, read Hotwire’s post on how they added additional criteria that are important to their business.

If you have a relatively mature optimization program with dedicated developer resources, you may be able to focus solely on tests based on impact, without needing to weigh effort. For inspiration, check out Hotwire’s binary scoring matrix to learn how they run over 120 tests a year.

3. Review your process

Once you’ve prioritized your ideas and run a few, set time aside to review how well your process works for your team. Below are two questions to consider.

Should you use a backlog or roadmap?

When you prioritize your testing ideas, you can put them into a backlog or a full, prioritized roadmap. The first option offers more flexibility but the second provides a stable cadence for organizing a complex workflow.

A prioritized backlog is just a queue. Once you’re done with one idea, you reach for the next. If you happen to chance upon an idea with great potential and you’d like to focus your efforts there for a while, you can.

With a full, prioritized roadmap, you commit to a timeline based on how long you think an experiment will run. Most mature programs prefer this approach, as it allows them to coordinate stakeholders and schedule a complex workflow.

With a full roadmap you plan more of your work in advance. You also build a regular cadence for incorporating insights and trends from completed tests and campaigns into a new round of testing. If you’d like to return to an idea, you can re-prioritize it and add a second iteration further down the line.

No matter which method you choose -- roadmap or backlog -- insights from completed tests and campaigns will help you re-prioritize in the next round.

Are you over or under-prioritizing?

As you get to know the cadence of your optimization program’s work cycle, evaluate whether you’re over-prioritizing (putting too many ducks in a row) or under-prioritizing your list.

Do you prioritize 25 ideas but execute just four or five before the next planning phase begins? You’re probably over-prioritizing. Teams that over-prioritize consistently fail to implement ideas lower in the prioritized roadmap. Evaluate whether low-priority hypotheses are worth prioritizing again and again. If fewer ideas would help you align your roadmap to your team’s cadence, consider capping the list at a lower number.

Do you feel you never have enough high-impact ideas? Teams that under-prioritize often run out of ideas before end of...
the cycle and find themselves returning to the ideation phase. If this is the case, consider focusing your efforts on generating more ideas for your backlog. An idea submission form and a business intelligence report can help you increase the number of high-quality hypotheses.

Automate idea submissions and ranking

Consider automating an idea submission process at your company. By automatically scoring the ideas submitted to your team, you’ll be able to evaluate and prioritize them more easily.

Create an idea submission form that asks questions about the resources and skill sets required for a given idea. The responses to this form populate a spreadsheet with built-in formulas that add or subtract points based on the responses. Voila! As soon as an idea is submitted, a score is generated based on your prioritization framework. Your ideas can be automatically sorted by those scores.

A formalized process of collecting and prioritizing ideas focuses your team on running strong tests and campaigns and pursuing huge wins that generate excitement for experimentation. Publishing your automated submission form company-wide can also help to democratize prioritization and spread awareness about optimization goals.

By equipping your team with a mechanism for outputting a well-prioritized list of tests and campaigns, you take a critical step towards building a sustainable and effective testing program.