



GoodData

8,563,983

# The Ultimate Guide To Embedded Analytics



This guide dives deep into the key considerations and strategies for successfully bringing customer facing analytics products to market. Understanding these best practices will also give you the tools you need to successfully implement a business intelligence solution within your organization.

# Table of Contents

Definitions and Concepts	4
Types of Analytics	8
The Keys to GTM	11
The Product Workshop	12
Setting the Project Goals	15
Know Your Audience	19
Determine the Requirements	24
Define the Product Offering	27
Pricing the Product	33
Establish Support	41
Launch Planning	51

# Foreword

Adding analytics to your product and creating dashboards that yield new performance insights is one of the most beneficial improvements you can make to your business. Collaborative social functions, new processes and workflows provided without the ability to view, analyze and constantly improve will place your business at a significant competitive disadvantage in today's data-driven world.

The good news is that platforms like GoodData allow you to extract data from a myriad of data sources, create metrics, charts, and dashboards that model your view of the world. If you're looking to deliver analytics to your customers, you can even choose to embed your analytical creations in software-as-a-service products or create company branded data portals.

The bad news is that it isn't enough to just have the analytics — you have to do something with them. You need to understand who will be using your data product, what problems they are trying to solve, what functionality they require, and how you can support a solution that meets all of these needs. You must build the product, price it, deliver it, and have the infrastructure handle any issues that arise. It sounds like a pretty daunting task doesn't it? Don't worry — it's really quite simple if you know the steps required and the issues that can arise. What you need is a plan...

This document is exactly that: a plan for taking the GoodData platform and building amazing data products that will solve mission-critical business problems for your users. It's a plan for structuring a product, for pricing the product, for launching the product, and for delivering support once the product is in production.

Let's get started!

# 1

## Definitions and Concepts

### Internal vs. External Project

#### **Internal or external?**

One of the first questions you need to ask yourself will be — is this an internal project or an external project? That is, are you implementing analytics to be used inside your organization (monitoring key performance indicators, tracking team progress, etc.) or are you adding analytical capabilities that will be used by your customers?

#### **Internal projects**

You might think that internal projects, those intended for use inside your organization, would be easier to implement. After all, this is for employees, not paying customers. You don't have to worry about the polish of the dashboards, the ease of use, the pricing, right? Wrong. While internal users can be more forgiving of missing metrics or report elements, they are also relying on the data to manage the business. They tend to know the data very well, to understand how metrics are related to one another, and are highly prone to ask questions and drill deeply into the data.

When implementing an internal project, consider treating it... exactly like a product deployment. Go through all the same steps from developing user personas to workflows to support processes, pricing and rollout planning.

### **Expert Tip:**

To maximize ROI, treat internal implementations like an external product launch.

Why would you go through all of this trouble just for an internal project? Your internal users are a captive audience, right? They really have to use the system, don't they? Possibly, but that doesn't mean they will like it or you will maximize the investment like you can if it is well thought-out and implemented. By treating an internal project just like a product that has to win users, allocate costs, support issues and training you will increase the probability of project success and create satisfied users.

### **External projects**

External projects are business intelligence implementations that are designed to create more value for your customers. Examples include embedding analytics into a inventory management product, adding dashboards to an online advertising campaign management application, giving reporting capabilities to the users of a CRM application, or even creating separate products to monetize expertise in digital marketing, service fulfillment and other industries.

When implementing analytics into products, factors not found during internal project come into play. Issues such as the following all must be considered:

- ▶ User login management
- ▶ Updating legal contracts
- ▶ Tracking usage
- ▶ Training customers
- ▶ Customized functionality

The rest of this go-to-market guide will help you through these issues and give you the keys you need to build a successful product using the GoodData platform.



## Types of systems

### **Types of BI tools**

One of the main problems with business intelligence implementation is exactly this: business intelligence means very different things to different people. When initiating your project ask various stakeholders what BI means to them.

#### **Key Point:**

Your team must have a common understanding of exactly what you mean when you say "business intelligence". Many times, they don't.

Chances are, you will get answers from "dashboards and charts" to "a system that predicts performance" to "a platform to store our data and generate reports." All of these are correct answers — the problems begin to occur when you don't start the project with a common understanding of what this business intelligence system will be for it's users. It's no fun to reach the end of a project and find out that the system you thought was successfully implemented doesn't meet the expectations of key stakeholders.

### **Visualization tools**

When you say the words "business intelligence" to people, visualization tools are what many of them immediately picture. These tools take data and turn it into charts and graphs that help users understand patterns and trends that would immediately be obvious by examining the data itself. With these tools, the graphs can be arranged into dashboards — collections of analytics that help tell a story, perhaps about the performance of a specific region of product line.

Visualization tools are cool; they create the eye-candy that make products look beautiful and fun to use. They provide interactivity so users can drill into data and learn more about specific data points.

But, they aren't the whole business intelligence story. You still need to get the data into the system, store the data, cleanse the data, and get it in the hands of the users. You need to ensure that the right charts are delivered to the right people, that the system can be easily marinated and configured, and that the system scales to terabytes of data and thousand (or millions) of users.

Without these critical elements, visualization tools might look nice but are not suitable candidates for use as the exclusive BI solution in a commercial product or a large-scale internal analytics system.

### **Exploratory tools**

Exploration tools are used — often by analysts — to dig into data sets and identify patterns and relationships. These tools can be quite sophisticated, allowing complex joins between data sources and the use of advanced statistical tools and modeling, but can be complicated for the average user. In most cases, pure exploration tools are best suited for use by advanced users who will be creating the datasets, metrics, and dashboard for others to use rather than for the average business user.

### **Data warehousing tools**

Data warehouse tools are the "heavy lifting" applications of the business intelligence world. Typically used to load, cleanse, and manipulate large quantities of data, these systems are absolutely vital to the preparation of data for analysis.

**"GoodData's end-to-end analytics platform has transformed how our customers leverage data and get fast answers to their business critical questions."**


- **Sid Shetty** President, Service Channel



# 2

## Types of Analytics

Another source of confusion that often arise when people speak of "implementing business intelligence" is exactly what kind of analytics they mean? Are they referring to showing patterns and trends, predicting future performance, or giving the user a remedy for poor performance? Each of these is an aspect of analytics, but it's critical to get your team in agreement on how you will be implementing analytics early in the project planning phase.



Category	Types of Analytics	Questions Answered
Prescriptive	<ul style="list-style-type: none"><li>▶ Optimization</li><li>▶ Randomized Testing</li></ul>	<ul style="list-style-type: none"><li>▶ What is the best that can happen?</li><li>▶ What happens if we try this?</li></ul>
Predictive	<ul style="list-style-type: none"><li>▶ Predictive modeling / forecasting</li><li>▶ Statistical modeling</li></ul>	<ul style="list-style-type: none"><li>▶ What will happen next?</li><li>▶ What is making this happen?</li></ul>
Diagnostic	<ul style="list-style-type: none"><li>▶ Data exploration</li><li>▶ Intuitive visuals</li></ul>	<ul style="list-style-type: none"><li>▶ Why did this happen?</li><li>▶ What insights can I gain?</li></ul>
Descriptive	<ul style="list-style-type: none"><li>▶ Alerts</li><li>▶ Query / drill down</li><li>▶ Ad Hoc reports / scorecards</li><li>▶ Standard reports</li></ul>	<ul style="list-style-type: none"><li>▶ What actions are needed?</li><li>▶ What is the problem?</li><li>▶ How many, often, where?</li><li>▶ What happened?</li></ul>

SOURCE  
Magic Quadrant for Business Intelligence and Analytics Platforms, February 5, 2013, Analysts: Kurt Schlegel, Rita L. Salem, Daniel Yuen, Jose Tapadinas

SOURCE  
Disambiguating Analytics, July 2, 2012, Sanjeev Kumar, International Institute for Analytics



As your analytics product matures and you discover which features are most desired by your users, you may augment the types of BI you offer, but garnering agreement on the initial go-to-market functionality is critical.

### **Descriptive Analytics**

Descriptive analytics are the most basic form of business intelligence. Here we are concerned with describing the situation as it currently exists, showing what happened, when it happened, allowing drill down into greater detail, and alerting the user when the issue occurs again through feature such as alerts.

### **Diagnostic Analytics**

One step up the business intelligence hierarchy is "diagnostic analytics". While descriptive analytics try to answer the question "what happened", diagnostic analytics are concerned with "why" something happened. Here, root cause analysis is key and the user is given functionality to drill deeply into data and see context to a problem situation. Was the hold time for a call center greater than expected? Drill down and find out that a new group of call-handlers started the day before and perhaps were not yet fully trained.

Diagnostic analytical systems don't give you all the answers, but allow you to explore the data, draw inferences, and make conclusions about why a problem may have occurred.

**Expert Tip:**  
Allowing users to easily drill into data is key to understanding root problems.

## **Predictive Analytics**

A further step up the business intelligence ladder is predictive analytics. Currently a very popular topic, predictive analytics asks (and hopefully answers) the question "what if"? What if I add three people to my help desk — will call times drop and by how much? What if we reduce our sales cycle by two days — will our quarterly revenue be impacted, and to what degree?

It's obvious why predictive analytics and the promise of showing the future are so in fashion these days, but it's important to remember that before you predict performance you need to understand the underlying patterns and root causes identified via descriptive and diagnostic tools.

## **Prescriptive Analytics**

Prescriptive analytics take it one step further and answer the question "so what should I do next?". These systems use information about what actions have worked well in the past, feed them back into the business intelligence system, and deliver recommendations about what course of action might be the most effective given the patterns or issues in the data. Obviously, prescriptive systems require a great deal of data, root cause information, and past results from actions taken in order to deliver their results. For this reason, prescriptive analytics— while a great target to aspire — are not the place to start your BI efforts.

# 3

## The Keys to GTM

The process for defining your the strategy around bringing your analytics product to market is crucial. The remainder of this guide will dive deep into each of the following categories to consider as you prepare to launch your analytics product:

1. **Hold a product strategy workshop:** Invite all stakeholders to a workshop to hash out the high level strategy.
2. **Set project goals:** Identify the MVP and establish an iterative model to deliver more value, faster.
3. **Know your audience:** Understand who they are, what they need and how they purchase in order to influence your product strategy.
4. **Map out product requirements:** Document the buyer personas and use cases to focus your development efforts.
5. **Define product structure and pricing:** Quite possibly the most critical aspect of your strategy - and most widely understood.
6. **Develop support ecosystem:** Ensure all of the right systems are in place to support and educate your customers.
7. **Prepare all teams for launch:** Make sure all teams from legal to support are well informed and have plenty of time to prepare for the release of your new product.

**"A good process can create discipline for the entire development, launch, and management cycle."**

- **ITSMA**, Go-to-Market Strategies: Eight Steps to Success



# 4

## The Product Workshop

The product workshop is the key to creating a successful analytics product. It is the foundation of a common understanding of the project goals and objectives; it is the point at which you decide exactly what the new product will be once implemented. It's possible that you will have a great product launch without conducting a product workshop, but the path to that day will be one filled with wrong turns, misunderstanding, and re-work. Completing the product workshop will ultimately save you time in the long run.

Let's dive into the mechanics of the workshop and learn why it's so critical to success...

### **Participants & Logistics**

A workshop can be held onsite at your office or offsite, whichever you think will yield the best result. If you have one of those offices where people will be called away for calls, emergencies, or meetings you might consider getting away from the distractions and conducting the session offsite.

It's not enough to hold a product workshop — you need to have the right participants in order to get the results you require. You cannot make important strategic decisions about your future analytics product without the right people in the room.

In most cases, the ideal size for the product team attending the workshop is 7-9 people with the following roles present:

1. **Product sponsor or champion:** This may be the CEO or it may be the head of product. The important thing is that it's the person with both the vision and the accountability to bring the new product into the world.
2. **Head of Sales:** This person will be accountable for selling the analytics and must be present to understand the capabilities to be offered.
3. **Head of Marketing:** The head of marketing should be present in order to understand the positioning of the product as well as it's target audience.
4. **Head of Engineering or Software Development:** As the development lead, this role must understand the data required for the analytics, how it will integrate with the overall product, and will help drive the development timeline.
5. **Project leader:** This person will be the day-to-day lead for the project, driving the timeline and ensuring all of the elements are coordinated.
6. **Facilitator:** The facilitator is essential to keep the product workshop on track and to prevent the team from getting bogged down on any single issue. We recommend that this be performed by a different person than the ones listed above to ensure that all viewpoints are given equal consideration.

The next three roles will support the product once launched and should attend the workshop so that they can raise any concerns prior to development:

1. Head of Operations or Support
2. Head of Finance
3. Head of Legal



## Agenda

As the product workshop is the official kick-off for the analytics implementation project, it is usually conducted after you have completed your search for a business intelligence vendor and signed the contract but before any technical work begins. The goal of the product workshop is simple: define the product goals, constraints, users, key functionality, pricing, and necessary support processes. All of these won't be completed during the workshop, but the requirements will be identified and the tasks assigned.

Below is a recommended agenda for a product workshop for an analytics project that will be embedded within an existing "software-as-a-service" (SaaS) product:

Agenda Item	Time Allocated
Project Overview	30 minutes
Set Project Goals	30 minutes
Define the Key User Personas	1 hour
Define the Product Structure	30 minutes
Set Product Table Stakes, Delighters and Boundaries	2 hours
Define the Pricing Model	1 hour
Determine Support Structure Implications	30 minutes
Set Project Timeline	30 minutes
<b>Total Workshop Time</b>	<b>6 hours</b>

Plan on discussing the overall outline for each of the topic areas and assigning tasks to be completed. Capture any side issues for later discussion and make sure that the team has reached consensus on decisions before moving on.

Ready to start diving into each part of the agenda? The next few chapters cover each item in detail.



# 5

## Setting Project Goals

### Project Goals

It's amazing how often people start projects without really developing a vision of where they would like to end up. The result is that it becomes very difficult to determine if you were successful, if more work needs to be done, and even if you are ready to launch.

The simple way to avoid this dilemma is by developing a set of goals for the analytics project at the very beginning. In your initial kick-off workshop, you should have the team brainstorm a set of project goals such as:

- ▶ Provision 25 customers by May 31st
- ▶ Provision 100 total customers by the end of the year
- ▶ Launch with an executive dashboard with descriptive analytics showing overall business performance by 3rd quarter
- ▶ Add in team lead dashboards with team performance (and drill-down to individual performance) by end of year

By setting goals at the outset of the project, you'll have a set of criteria around which you can rally the team and then celebrate once you've achieved your objectives.

## **Project Constraints**

Constraints are as important to the success of the project as the goals, and can even help you think of creative ways of solving problems. After you've determined the project's goals, start listing out the boundaries. These are the things that are limits or restrictions for your project. For example:

- ▶ Only use our Engineers after their work for the week is completed
- ▶ Make the dashboards look & feel like our core brand
- ▶ Do not use any data from the CRM or financial systems
- ▶ Comply with HIPAA regulations
- ▶ Get permission from each customer before using their data for benchmarking

The constraints, like the goals, help you frame the project and understand what - although it might be nice to do - is off-limits.

### **Fact:**

Scope creep is the #1 reason why projects lose steam.

## **Table Stakes & Delighters**

In addition to setting the goals for the project, it's important to determine which functionality is absolutely essential and which is nice to have, but not required for product launch. We call the "must-have" functionality "table stakes" and the "our customer would love it, but it's not required functionality" is called a "delighter".

By dividing functionality into these categories early on in the project, you can rally the team around completing the list of table stakes without getting bogged down on the details of the delighters.



Here's an example of table stakes and delighters:

**Table Stakes:**

- ▶ Single-sign on
- ▶ Embedded within our core app
- ▶ Executive dashboard
- ▶ Drill down to regions

**Delighters:**

- ▶ Alerts when key thresholds are exceeded
- ▶ Ability to annotate charts
- ▶ System auto-filters the dashboard to show "my data"

Once you've got the list of table stakes and delighters developed, refer to it when people suggest changes or fixes. "That's a great idea, but it's a delighter, not a table stake. Let's get the table stakes done first, then move on to the delighters". This can help ensure the team is working first on critical functionality required for launch before worrying about the nice-to-have's.

**The minimum viable product**

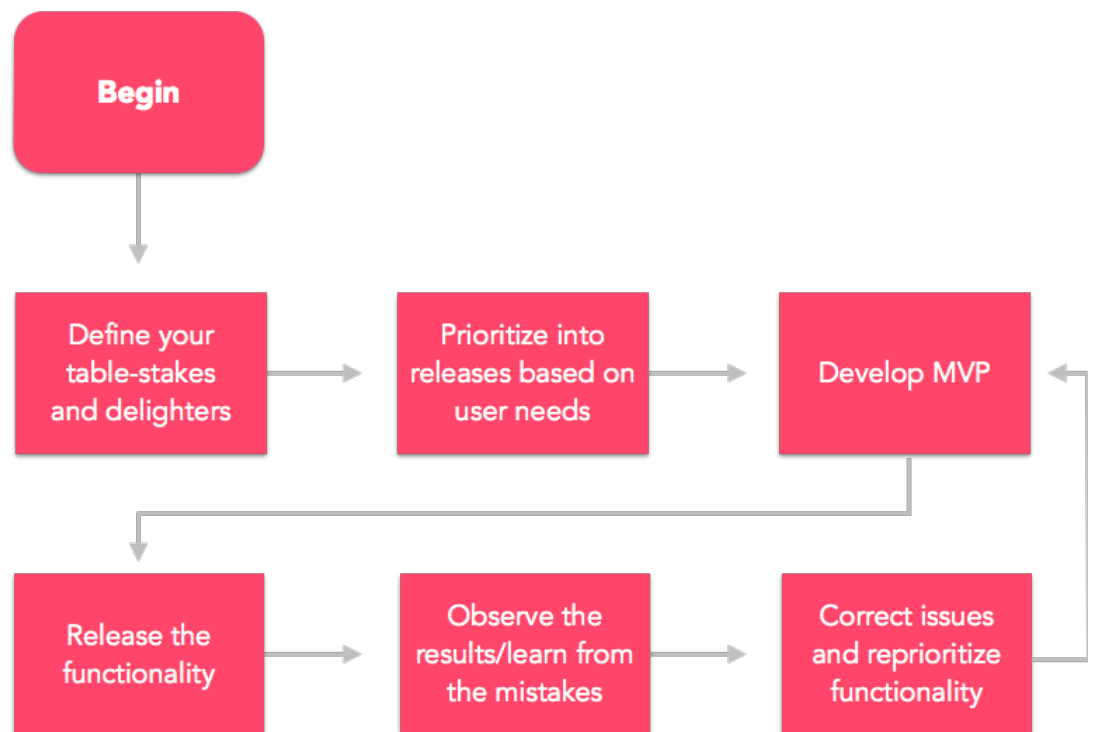
How are you going to deliver all of these great features? Will you hold off on launching until you can deliver every one of your table-stakes and delighters on day one? History shows that this is a bad idea. Often by the time you get your perfected product to market, the market has changed and you are back to square one. A better strategy is that of the minimum viable product or "MVP".

The MVP concept is simple: design the simplest, most basic functionality that will get the job done and iterate from there. This means that you don't wait for a "perfect" product before releasing to production.



You go to market with the bare essentials required to meet some of the needs of some of the users and iterate from there using the lessons you've learned to improve the product with each release.

The key with the MVP design is that you need to get a new functionality out, up, and running as quickly as possible to test its performance in the real world. Those super-complex, "perfect" systems will need to reach the real-world stage at some point — wouldn't you rather have spent two-thirds less time in process design when you find out that your process has major flaws that must be corrected? Use the MVP as your initial test platform to challenge your assumptions and ideas about your new product and use what you learn in the next iteration.





# Know Your Audience

## Know the users

An often overlooked element of creating a great analytical product is gaining a thorough understanding of your users. You might be thinking — "wait, I know my users. They are the people that will be logging in each day to use the system". True, but this isn't the entire story.

Your users may span a wide range from senior executives to front-line employees, from customers to suppliers, from engineers to sales representatives. Each of these types of users will have unique needs that must be addressed. For example, the Chief Marketing Officer using your analytical application may have questions such as:

- ▶ Where are we spending our budget?
- ▶ Are our resources being used efficiently?
- ▶ Where can I intervene most effectively so that I can increase our conversion rate?

These are broad, strategic questions that may require the application to pull from multiple data sets and show aggregated trends and patterns.

In contrast, a front-line employee such as a call center representative may need answers to the following:

- ▶ What's my close rate for calls?
- ▶ What's my average time on a call?
- ▶ Am I trending ahead of or behind my goals?
- ▶ What products take the most time on each call?

Here the questions are much more tailored to an individual and likely come from a single data set.

Now pretend that you hadn't spent the time defining your user personas. You might have kicked-off your efforts by gathering people from the executive team, sales, marketing, product, and engineering in a room and brainstorming a giant list of possible metrics to display. And, you might have come up with the right answer in many cases. But, would you have organized the metrics properly for each user type or would you have mixed high-level executive metrics with detailed front-line employee metrics? Experience shows us that failing to define users at the outset greatly increases the likelihood that you'll miss key metrics and that you'll fail to organize the analytics in the most effective way for each user group.

Here's how you can avoid these pitfalls...

### **Select personas**

The first step is to create a list of the personas that you will serve with your analytical application.

Gather key stakeholders in a room and develop a list such as the following:

**Personas for analytics within our SaaS Product:**

- ▶ The customer's Chief Marketing Officer
- ▶ The customer's Sales VP
- ▶ The customer's Product Management VP
- ▶ The customer's product managers
- ▶ The customer's marketing managers
- ▶ The customer's team leads
- ▶ The customer's sales reps
- ▶ Our own operations team (to monitor customer performance)
- ▶ Our own account team (to help customers)
- ▶ Our SaaS operations team (to monitor web application performance)
- ▶ Our finance team (to track billings)
- ▶ etc.

**Expert Tip:**

Prioritizing your target personas based on value to the user and complexity of the solution will keep your strategy targeted and focus feature development.

You'll likely end up with a lengthy list — one that is too much to pursue for your first iteration of the system. With the help of your product team, pick 2-3 personas that you would like to focus on first. Select the personas by balancing the value the user will get from the system with the complexity involved in delivering the functionality that they will require. The Chief Marketing Officer may get a lot of value from the system, but if the functionality that person requires is extremely deep and complicated to deliver, you might want to start with a more simple persona.

However, you don't want to discard the personas that aren't served with the initial launch of the product.



Create a timeline that shows the priority and timing when you'll address each persona and use this timeline to drive your analytics roadmap. By doing this, you'll ensure that the features you plan for development are tied directly to a persona's need.

### **Develop the persona**

Once you've selected the personas to be addressed by your product, the next step is to create the detailed persona map — the "bio" that describes the persona's needs. Here's a simple flow that will help you develop a persona:

1. **Start with the basic details.** Give the persona a name and a title for their position. These items help personalize the persona and allow you to better empathize with the goals and needs that you'll be describing later on.
2. **List the key characteristics for this persona.** Are they a leader? Are they looked to as an authority? Do they have to jump from crisis to crisis rapidly or do they have time to focus on an issue?
3. **List the frustrations and pain points this user has.** It's easy to get derailed here and list every problem the persona might encounter, but try to focus the list to the pains the person encounters while trying to perform their job lacking the analytics you are going to provide.
4. **Identify the key questions the persona is asking on a regular basis.** These might be the items they routinely request during staff meetings or things they request as reports on a regular basis.
5. **List the applications the persona might use** to get their answers today lacking the analytics you'll be implementing.
6. **Create a list of the features the user might want to see.** Don't spend too much time here — the questions they have are more important — but if there are special "table stake" features the persona needs to have, list them.

Once you've got the persona fleshed out, create a "Persona Bio" like the example below. As you work through your project, refer back to the persona bio frequently so that you don't lose sight of the "person" for who you are creating the analytics.

Jon



<i>Role or Title:</i>	<i>VP Customer Advocacy</i>	<i>Key Characteristics</i>
<p><b><i>“We need to know if customers are unhappy with our service, what aspects of our service, and what we can do to improve performance.”</i></b></p>		<ul style="list-style-type: none"> <li>• Looked to for answers</li> <li>• Is a trusted advisor to the Sales team</li> <li>• Focused on root cause analysis</li> <li>• Uses NPS to track performance</li> </ul>

### ***Frustrations and Pain Points***

- I feel like I'm operating blindly until the quarterly reviews occur
- I have to rely on others to build reports to tell me what's going on
- I focus on individual customers & issues that seem to be "hot" that day — I don't know if these are the right things
- I worry that I'm not making any long-term improvements, just day-to-day tactical fixes
- It takes us way too long to identify trends and patterns in customer usage. By the time we see them, it's too late..

<i>Questions</i>	<i>Other applications</i>	<i>Feature requests</i>
<ul style="list-style-type: none"> <li>• Is customer health trending up or down and why?</li> <li>• Are there global issues that I can correct to improve retention?</li> <li>• If I fix the issues, how much uplift do I get?</li> <li>• How do we compare to everyone else for retention and customer satisfaction?</li> </ul>	<ul style="list-style-type: none"> <li>• SurveyMonkey</li> <li>• MS Excel</li> <li>• NPS tools (Verint, Vovici)</li> <li>• Tableau</li> </ul>	<ul style="list-style-type: none"> <li>• Allow customers with high value and potential issues to be made easily identifiable</li> <li>• Show trends over time globally, for segments, products, and specific customers.</li> <li>• Show how I perform relative to peers</li> <li>• Allow me to track corrective actions made and the resultant performance shifts.</li> </ul>

# 7

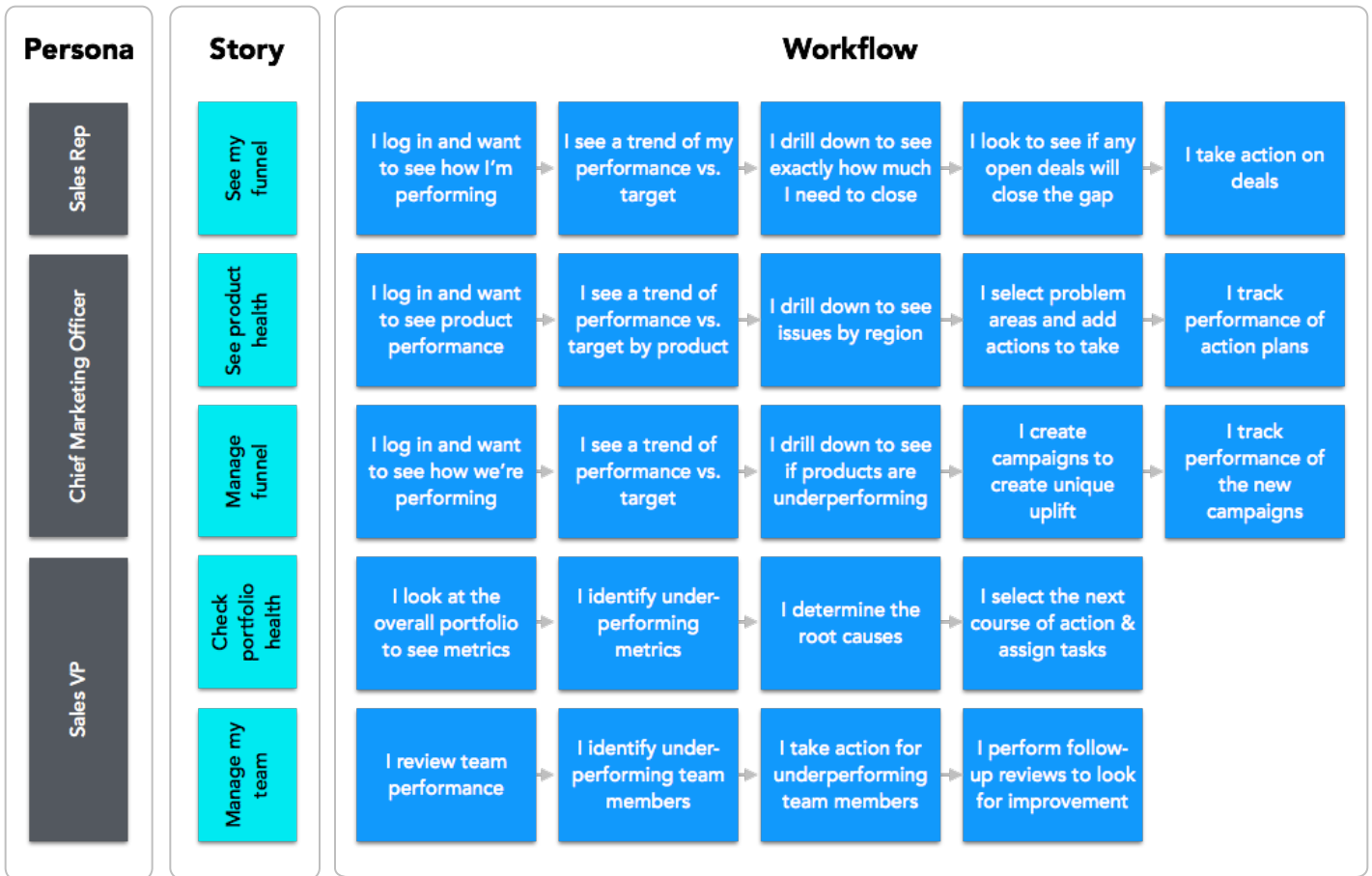
## Determine the Requirements

### Define workflows

Now that you've got the personas identified and prioritized and the bios developed, the next step is to determine what each persona needs in order to perform their job. We do this by mapping out the common workflows for the persona. Here's how it's done:

1. **Brainstorm a list of the top 5 functions the persona performs on a regular basis.** Example of these functions are: review team performance, identify campaigns that are underperforming, and predict revenue from products.
2. **Give each of these functions to be performed a "story" title** such as "Review Performance" or "See Product Health".
3. **For each story that you created, build a simple 5-7 step flowchart** that describes the steps the persona might perform in the future, using the analytics to accomplish the story. For example, to check product health the user logs in, sees a trend chart, clicks the chart to drill down on a problem area, decides which actions to take, etc.





Once you've got the workflows mapped out, the next step is to determine the analytics required to complete each workflow.

### Determine the analytics required

After you've defined the personas, their needs, and workflows they are performing on a regular basis, the next step is to figure out which analytics will help fill the gaps that they're currently experiencing. Take the workflow for each persona and go through it looking for analytics that might be of help. If the persona has a workflow called "review team performance" a dashboard that shows each team member's performance all side-by-side might be of use.

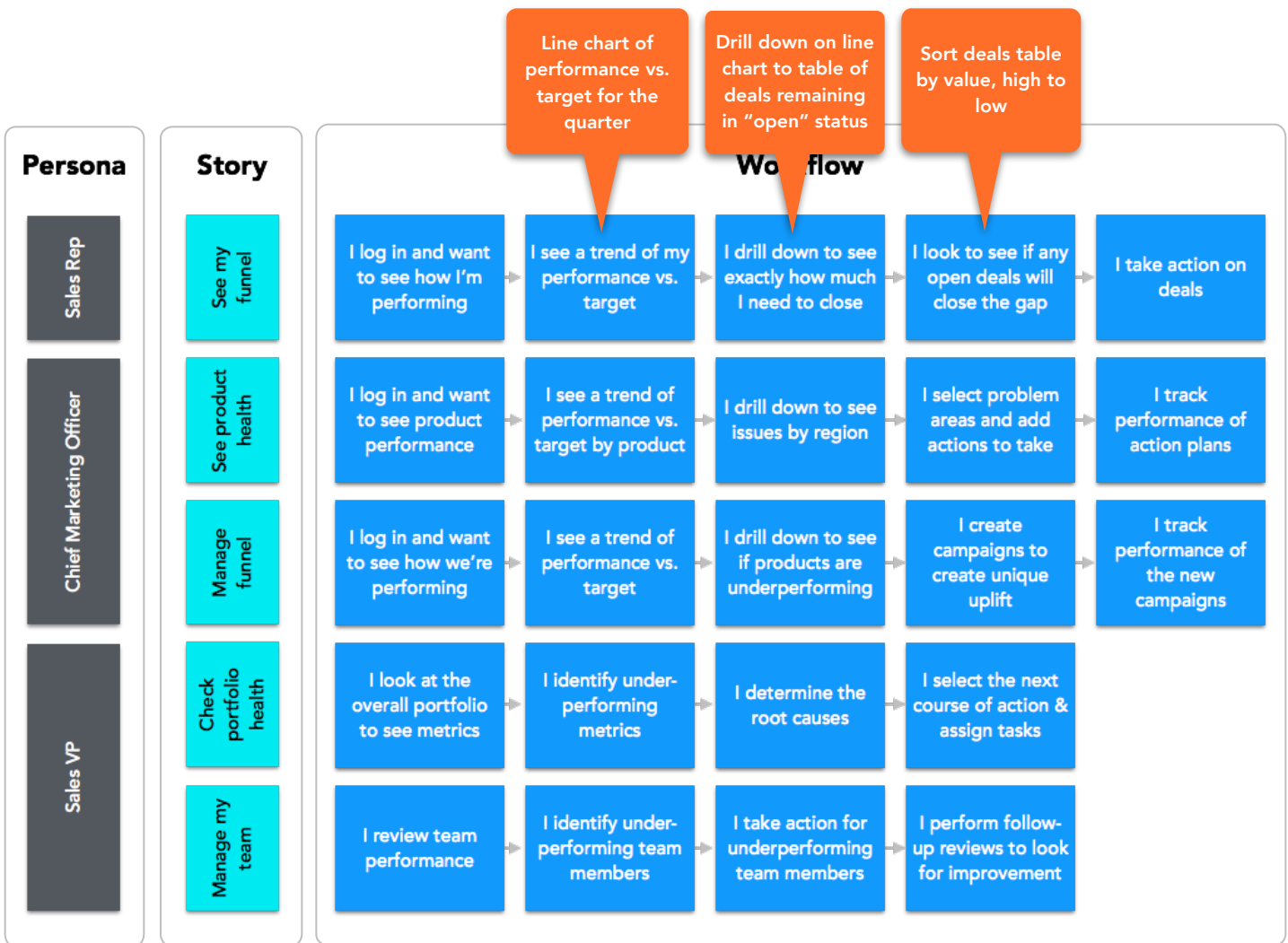
If the user has a workflow called "review product performance", a dashboard that shows them the revenue associated with each product broken up by region, product line, new users versus existing users etc. might be helpful.



What you're trying to do is ensure that every analysis you place on every dashboard is directly tied back to a workflow required by a persona.

You're trying to avoid extraneous analytics that serves no purpose and make sure that you haven't left any gaps or missed any pain points experienced by potential users.

We recommend annotating the mini flowcharts you've developed showing the workflows for each persona with the analytics that will solve the pain points within each workflow:



# 8

## Define the Product Offering

### Product Structure

#### **The offer structure overview**

You've set goals for your project, defined your personas, and determined the analytics they require but, what does the product really include? Does everyone get every feature? Do you have different levels of service? Here are some options and ways to decide what makes the most sense for your situation.

#### **The "all in" product model**

The simplest model is the "all in" product structure. In this scenario, every user that has access to the application has access to the same analytical capabilities. Calculating the number of users and the expected revenue is easy: count the number of logins for the product and that's how many analytics users you will have. Multiply that by the price you'll charge for the analytics and you've now got your anticipated additional revenue.

The problem with this model is that it constrains what you are able to offer in the future. What if you add advanced functionality that increases your platform costs? Do you include this for all users or hold it back to just a subset of roles? What if the customer wants specialized functionality or more data stored? How will you price that request?

Although it's a great way to start out (and is very often the initial go-to-market model for analytical products), the second option—tiered functionality—gives you more flexibility.

## The tiered product model

You are probably already familiar with the tiered product model and have likely used in the past when you've purchased software-as-a-service products. This model breaks functionality into multiple tiers or levels, each higher tier with added functionality and an increased price. The tiers are usually called something like "Basic, Plus, and Pro" or "Standard, Premium, and Enterprise". It doesn't really matter what you call your tiers, what matters is what you include in each level to make a compelling case for a customer to move from the lower priced levels to the higher priced options.

## Creating product tiers

Once you've decided that a tiered product offering is the direction you'd like to take, you need to decide exactly what the customer gets in each product tier. Product teams usually use a combination of features, connectivity, and data volumes stored to construct tiers that are most useful for customers. Below is a list of categories that are commonly used to distinguish between product tiers.

Category	Details	Typical Options
Years of data available	The number of years of information available to be presented to the user	1 year; 3 years; 5 years
Benchmarking	Allow comparisons to aggregated and anonymized data from others	No benchmarking; internal; external
Connections to external data sources	Allow connections to outside data sources (e.g. salesforce.com)	No connections; connection to 1 system; connections to 3 systems
Ad hoc analytics	Allow users to create new charts and other analytics using existing metrics	No ad hoc; ad hoc allowed
Custom metrics	Create non-standard metrics still supported by the standard model	Standard only; modify/create new metrics
Custom model	Allow the user to have a custom model, metrics, dimensions	Standard model; model customized with metrics and dimensions
Drill-down levels	Allow drill down to record level data	Drill down 1-2 levels; drill down to record level data



This list is by no means comprehensive, but should serve as a good starting point for a discussion within the product team. Start by making a list of the functionality you would like to offer.

This method makes it easy to visualize how the various tiers compare to one another. As you perform this exercise, ask yourself:

- ▶ Is there a compelling reason for a customer to move from the lowest tier to a higher tier?
- ▶ Do the features we've included in the higher tiers add additional costs for us to provide? If so, are those features placed in a tier where we can charge enough of a premium to cover the costs.
- ▶ What percentage of customers do you think would stay at the base tier vs. upgrading to higher tiers?
- ▶ Will the basic function peak the interest of the average user, making them interested in more advanced functionality?
- ▶ How does this offering compare to competitive products?

When you've completed the tiering exercise, you should have a model that looks like the example below:

<b>Standard</b>	<b>Plus</b>	<b>Pro</b>
<ul style="list-style-type: none"><li>▶ Included for all customers</li><li>▶ Uses standard project / template</li><li>▶ No customization</li><li>▶ Users cannot modify dashboards</li><li>▶ No internal or external benchmarking</li><li>▶ No external data sources</li><li>▶ 1 yr. historical data</li></ul>	<ul style="list-style-type: none"><li>▶ Upgrade from Standard</li><li>▶ Uses Standard project / template</li><li>▶ Customization for an additional fee</li><li>▶ Users cannot modify dashboards</li><li>▶ Internal benchmarking included</li><li>▶ Connection to 1 data source (hourly rate)</li><li>▶ 2 yr. historical data</li></ul>	<ul style="list-style-type: none"><li>▶ Upgrade from Standard / Plus</li><li>▶ Uses standard project and is customized per customer needs</li><li>▶ Users can modify dashboards</li><li>▶ Includes external benchmarking</li><li>▶ Connection to 3 data sources (hourly rate)</li><li>▶ 3 yr. historical data</li></ul>

## Product boundaries

After you've decided what will be in the product, it's just as important to decide what won't be part of the product offering. It's often very tempting, given a customer with a request and a budget to spend, to agree to deviate from your product vision and add a little feature here, a small change there. Although it can seem so small and harmless at the time, making these little changes to the analytical product to support specific, unique customer requests can have a far reaching impact. You need to decide early on in the process what types of requests you'll support and which you'll turn down.

We call these the product boundaries.

Product boundaries are critical to creating a great product because they determine the resources you will have available to support the product for the entire customer base. Here's an example: let's say you've got a product with a dashboard and several charts. Every customer has the same layout, the same metrics, and the same charts. Along comes a new customer who is requesting something different. Instead of connecting to the data source that everyone else is using, this customer want you connect to a home-grown, non-standard database that you've never worked with before. They're willing to pay — are you willing to support this configuration? If you say yes, consider the impact to the support team. When a call comes in, will they know how to address the situation given the non-standard nature of the configuration? What about the operations group? Will they be able to track the unique costs associated with the one-off customer? You may decide that it's worth it to pursue opportunities such as this, but you need to consider the implications first and set clear boundaries for what you will and won't allow.

We recommend dividing functionality into the following groups:

1. **Features and functionality supported as part of the core product:** Of course we'll do this — it's part of the product!
2. **Features we'll support for an extra fee:** We'll do it, but you need to pay extra. Things such as connecting to different data sources or adding new charts and metrics might fall into this category.
3. **Things we won't do:** Connecting to home-grown, on-premise data sources, real-time data refreshes, or regionalizing into a foreign language may fall into this grouping.

The specifics of the features/functions you won't support don't really matter. What is critical is that you set boundaries and agree not to cross them even in the face of tantalizing new sales deal.

### **The offer presentation**

As important as the product structure is the product presentation. That is, how will users get to the analytics you'll be offering? When considering the product presentation, there are three areas that you'll need to think through:

1. Access to the analytics
2. White labeling
3. Contracts

### **Access to analytics**

First, how will users be accessing the analytics you're offering? Will they be logging directly into the platform (i.e. logging into a branded version of GoodData directly with a username and password) or will they be accessing the analytics from within your product? In most cases, product teams choose to embed the analytics within their product so that it feels like the core product itself.



If this is the case with your system, you'll need to account for the time required to set up the single sign-on system that allows users logged into your product to also be logged into the analytical platform, the time for your user experience team to ensure that the analytics match the overall design of the product, and the necessary time to test the embedded functionality.

### **White-labeling**

If you are embedding the analytics within your product, you then need to decide — are the analytics white-labeled or black-labeled?

In a white-labeled scenario, the underlying analytics platform provider is never revealed to the end user. All labeling, user messaging, and error messaging is either suppressed or stripped of text mentioning the 3rd party providing the analytics. In a black-label situation, the analytics platform is clearly disclosed to the end user (e.g. "Powered By GoodData").

Items to consider when choosing between the two options include parity between the core application and the analytics for:

- ▶ Filter options
- ▶ Dimension options
- ▶ Data refresh frequency
- ▶ Regionalization/localization
- ▶ Import/export capabilities

If you do have parity between the analytics and the rest of the product (or feel any differences are minor), then white-label might be a viable choice. If you have differences that might create questions or a jarring experience ("why is the application showing 5 years of data here, but 1 year over here?") then black-labeling might help avoid confusion for your users.



## Pricing the product

Pricing is one of the trickiest parts of going-to-market with the analytical product and as a result, many teams put this key decision off until late in the process. This is a big mistake. Pricing and product functionality have a symbiotic relationship — small changes to one often equate to necessary changes to the other. They can't be addressed in isolation. Want to store more data? You might want to increase your price to account for this. Want to offer the base tier of analytics free of charge? Then don't include all the functionality you've got or users will never upgrade. It's best that you think through the manner in which you'd like to price the product early so that you can revise the functionality tiers as necessary.

Should you charge?

Should you charge users for the new analytical capabilities that you are offering? This question comes up time and again but is one of the simplest to answer:

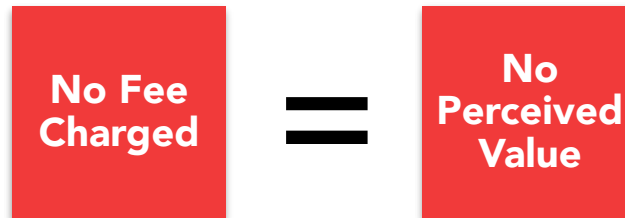
***Yes. You should charge customers for the analytics.***

It's not the answer some people want to hear — "but", they say, "it will help us compete in the market. The cost to provide it is low. We should give it to customers free of charge.”:

***No. You should charge for the functionality.***

The reason for charging is twofold: First, you are adding value to the product and customers should pay a fee for added value. Getting customers used to receiving new, highly valuable capabilities without an additional charge is a dangerous path to start down. Later on, if you add predictive analytics or prescriptive functionality — will you charge for it? If you gave the analytics away free at first, charging for new capabilities may be a more difficult sell.

The second reason you need to charge for the analytics is that it causes you to treat the analytics seriously. By making the analytics a paid, integral part of your product offering, it forces you to think through the analytics in detail. You can't dismiss the analytical elements saying "well, users don't pay for that anyway". Charging for the analytics forces you to create a compelling product for the market.

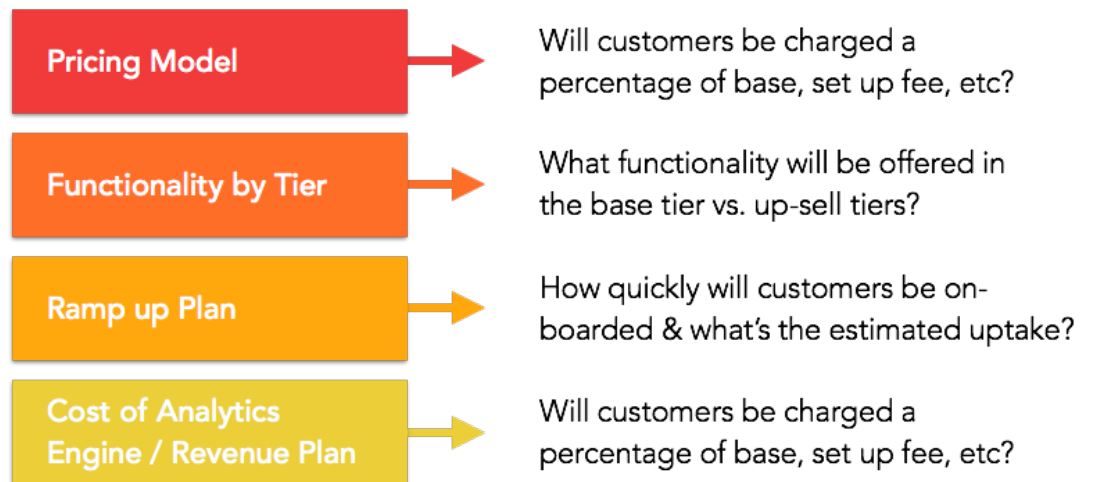


### Considerations when pricing

So how do you determine pricing for your analytical product? A good place to start is by considering the following elements:

1. **Pricing model:** How do you want to base your pricing? Will it be a flat fee? A percentage of the transaction fees?
2. **Functionality Tiers:** Will everyone get the same analytical functionality or will you have tiers to which a user can upgrade and receive greater capabilities? If you are planning on offering tiers of service, you'll want cover your costs but still make it easy and enticing for customers to move to the next level and receive the added capabilities.
3. **Ramp up:** Are you planning to slowly add the analytical capabilities for users or will you make the functionality available to all users quickly? If you plan to ramp slowly, you might be able to grow the platform costs slowly as well giving you more flexibility in the pricing you can extend to customers.

4. **Cost of the Platform:** You'll want to cover your platform costs with your product pricing, but you need to understand those costs and how they interact with the capabilities you'll be offering to your customers. If data storage is the primary cost driver for your solution, consider offering a basic tier that has either limited data or highly aggregated data (averages or monthly values instead of daily values) so that you can control costs and offer a lower price. If customers need large volumes of data, create a "large data surcharge" that covers your costs but still allows them the features they require.



### **Pricing models**

Once you've got a good understanding of your cost drivers, your ramp plan, and how you would like to go to market you are ready to choose a pricing model.

Assuming that you already have a product and you are looking to add analytical functionality, there are three ways that you can price the added analytical functionality.

<p><b>Option A:</b></p> <p>Charge a percentage increase over the base price of the product.</p>	<p>This option works well when you have a relatively costly product and the percentage increase you charge to cover your costs plus margin can be small. If you have a product for which you would normally charge \$5M per year, a 1% fee increase for the new analytical capabilities may look tiny in comparison to the functionality delivered.</p> <p>The drawback to this option is that it doesn't work well for situations where the core product has a low price point and volume of customers. If the product is normally priced at \$10K per year, you might have to charge an additional 50% to cover analytics costs. To most customers, an increase of this percentage would require a serious conversation.</p>
<p><b>Option B:</b></p> <p>Increase per transaction fees.</p>	<p>This option works well if your product pricing is based on the number of transactions conducted, such as an ordering system. If you are processing high volumes of transactions, adding a small surcharge to each transaction might be enough to cover your costs while still matching the model that your customers are familiar with.</p> <p>However, if the transaction volume is low this model may not work well as the cost added to each transaction might be larger and become concerning to customers.</p>
<p><b>Option C:</b></p> <p>Add a flat line item charge to the product cost</p>	<p>A flat charge for analytics added to a customer's contract may seem like the logical choice, but often makes the sales process more difficult. In many cases, each line item of a contract becomes a negotiating point during the initial sales process. When the customer sees a line item detailing "\$5,000 for Analytics Functionality," they may start asking questions such as "do we really need this?" or "why don't you just include this functionality?" In the worst case, the customer may ask that you remove the analytical functionality and the charge eliminating your ability to show the value that you have created through your business intelligence implementation.</p> <p>If you are concerned that the line item charge may become a point of contention during negotiations, it might be better to select one of the other options.</p>

Below are examples of each pricing option:

	Details	Example
<b>A</b>	Charge a percentage increase from the base price of the core product for all users	Product A costs 1% of assets managed. Add analytics and increase fee to 1.25%
<b>B</b>	Increase transaction fee percentage charged for all users	Product B charges users \$0.05 per transaction. Add analytics and charge \$0.06 per transaction
<b>C</b>	Add a flat line item charge to the cost of the product	Product C costs \$1000 per year. When purchasing, the user can add BI for an extra \$100 per year

### How much to charge?

You've determined that yes, you should charge for your new analytical capabilities and determined the method for charging a fee, such as a percentage or a flat-rate. But the question still remains: Exactly how much do I charge for the analytics?

When setting the price for your analytics, three goals should be at the forefront of your conversations:

1. Recover your costs to provide the analytical capabilities
2. Signal the value of the analytics while still encouraging usage
3. Earn enough to cover future development and other product goals

The first goal is obvious — you want to charge enough so that you are not losing money on each new customer that you bring on to your analytics platform. This implies that the price you charge will be at least your total cost per month divided by the customer count plus a percentage of the costs you incurred during implementation:

$$\frac{\text{Monthly Cost to Provide Analytics}}{\text{Number of Customers}} + \text{Percentage of Implementation costs} = \text{Monthly Fee to Charge for Analytics}$$



The problems occur when the resulting fee would be too high for customers to bear or too low to indicate that the analytics are providing any real value. Let's deal with the "too low" scenario first.

### Scenario #1

If in your cost equation is the following:

$$\frac{(\$25,000/\text{month platform fees})}{1,000 \text{ customer accounts}} + \text{Percentage of Implementation costs} = \$25/\text{month} + \text{percentage of initial costs}$$

You would charge \$25 plus some percentage of your initial project implementation costs to your customers each month for the new analytical capabilities. Depending on the overall pricing structure of your product, you may decide that this is too low to indicate real value derived from the analytics.

If you normally charge each customer account \$10,000/month in service fees, an additional \$25 is a drop in the bucket. They likely won't notice this trivial increase and it certainly won't indicate to them that they've receive tremendous new capabilities, violating the second of the three goals.

In a situation such as this, consider increasing the product cost by 10-15% (+\$1,000 to +\$1,500/month) for the analytics instead of tying the fees directly to the costs. This will cover costs, signal value, and meet the third goal of providing operating funds to continue development of analytical capabilities.

**“You owe it to yourself and to your business to be relentless in managing your product pricing. Remember, how you set the price of the products could be the difference between the success -- or failure -- of your business.”**

- Elizabeth Wasserman, How to Price Your Products, INC.



## Scenario #2

In the other scenario, you might have a situation such as:

$$\frac{(\$25,000/\text{month platform fees})}{100 \text{ customer accounts}} + \text{Percentage of Implementation costs} = \$250/\text{month} + \text{percentage of initial costs}$$

In this situation if your product normally costs each user \$500 per month, a 50% increase in the price might seem excessive. In a scenario such as this, you will need to either increase your overall customer count, reduce your platform fees by limiting functionality, data stored, etc., or plan on spending more time in the sales process explaining the significant value that you've added to your product.

Finally, what about the second half of these equations — that part that says "percentage of implementation costs"? A good heuristic is to try to recover 50% of your implementation costs in the first year through your pricing model. Therefore, if your situation was the same as the first scenario above and you spent \$200,000 in implementation fees:

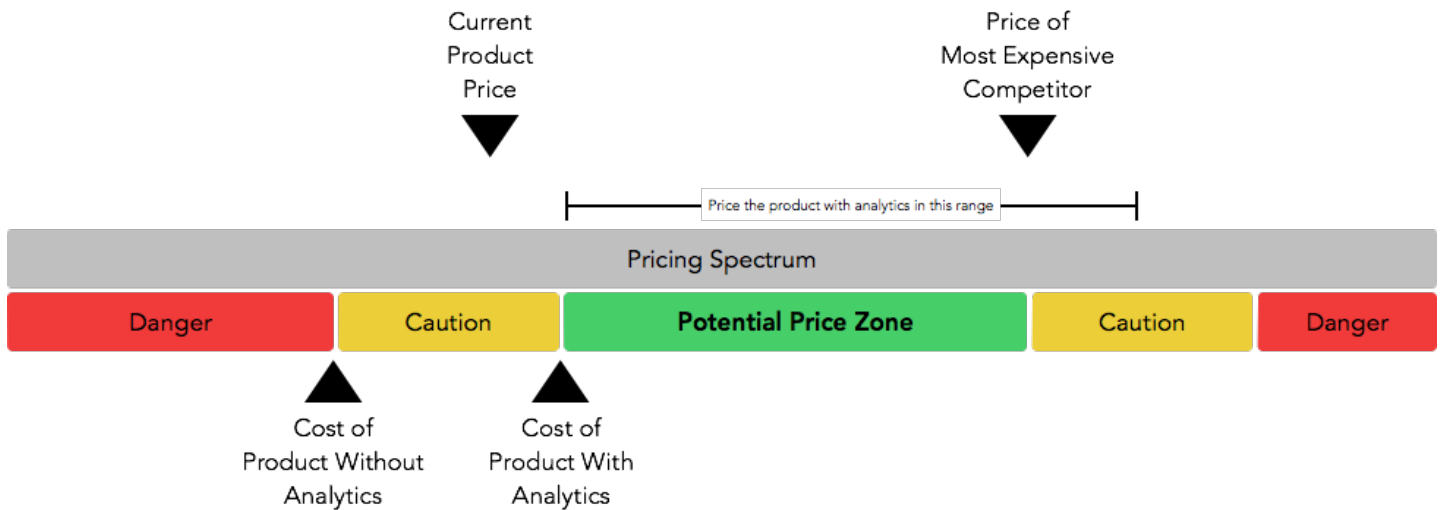
$$\frac{(\$25,000/\text{month platform fees})}{1,000 \text{ customer accounts}} + \frac{(\$200,000 * 50\%)/12}{1,000 \text{ customer accounts}} = \$25/\text{month} + \$8.33 \text{ (the percentage of initial costs)}$$

At \$33 per month you've hit your 50% of the implementation costs and your price is still way too low — you could increase the price to cover all of your implementation fees in year one and likely still be at an acceptable price point.

Again, the goal is to cover your costs while making the analytics attractive to customers and signaling that you've added new value to the product. If your pricing equation doesn't work out immediately, adjust your costs, implementation cost recovery speed, and profit to be made until you find a solution.



Another way to think about pricing is by using the Pricing Spectrum:



The pricing spectrum helps you think about your price in the context of the four key drivers that determine the range in which you should charge: cost to deliver the product today without analytics, cost to deliver the product with analytics included, current product pricing without analytics, and the price of your most expensive but comparable competitor.

Pricing the product below the cost to deliver it (cost of goods sold) puts you in the caution or danger zone — not where you want to be. Similarly, pricing significantly above the price of your most expensive competitor can make the sales process a challenge. You'll spend lots of time educating the customer on why your product is so much better than the competition that it warrants a premium. The ideal range is shown by the green section of the spectrum; above cost, above to price of the product today, but near the price of the competition.



# 9

## Establish Support

### Develop the Support Ecosystem

#### **Brainstorm potential process impacts**

The best way to quickly develop a list of potential processes and functions that may be impacted by your new business intelligence system is to gather potential stakeholders in a room and start listing out the possibilities in a brainstorming session. Conduct a one hour workshop with members from operations, finance, engineering, marketing, sales, and support all present. Keep it short, but conduct a round-robin session where as you go around the table, each person lists off a single process that may need to be revisited in order to support the analytics product. Once that list is complete, start working on trimming the list down to the processes that are most impacted and which need to be examined.

Examples of processes that may be impacted include:

- ▶ The sales process
- ▶ The contracting/negotiation process
- ▶ The new account setup process
- ▶ The customer support process
- ▶ The training process

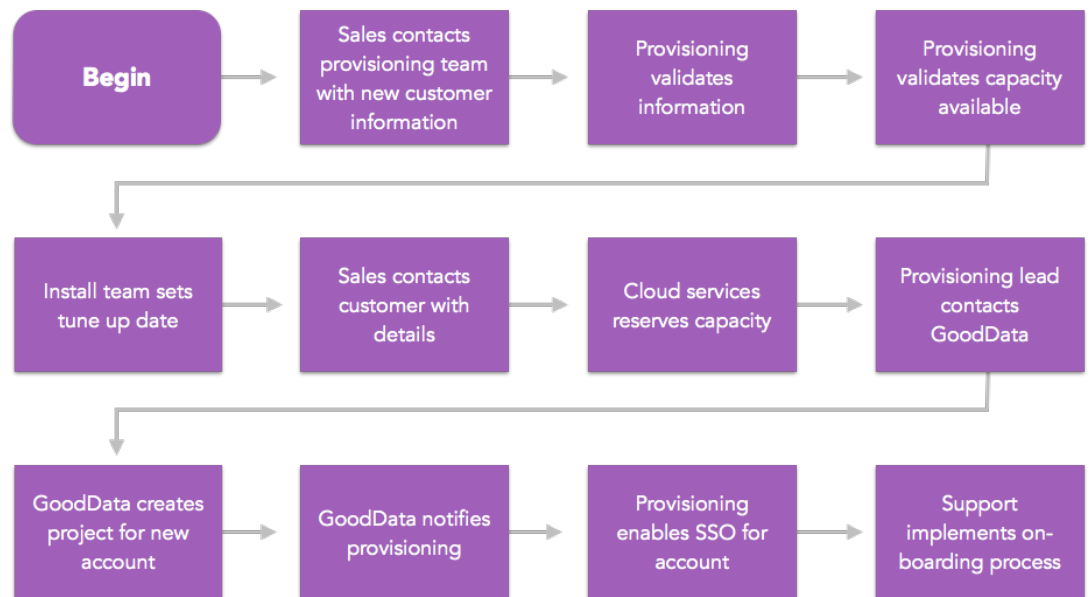
#### **Flowchart key processes**

After brainstorming the list of processes that may be impacted by the implementation of your business intelligence platform, the next step is to create a flowchart showing the specific steps within each process.

This flowchart doesn't have to be pages long with thousands of steps, a high-level flowchart is fine as long as it gives enough information to show who is doing what step at what time.

The key here is to document the process with enough detail that you give everyone involved an understanding of what will be happening, have enough context to test your process before you go live, and that you can use the document to help new employees understand the activities to be performed. Below is a simplified example of a new customer order process:

### Test the Processes



Once you've determined all the processes you'll need to support with your new business intelligence platform, it's critical that you test out each of these processes before it's put into action for real. There are three techniques that you can use to test your processes and make sure you've covered all your bases.

- ▶ The process walk-through
- ▶ The Failure Modes and Effects chart
- ▶ The RACI chart



## **Walking the Process**

Once you have the flowchart created, it's time to start testing to make sure you've adequately captured what will happen in the process. Often while testing, you will find that you overlooked a step or a key hand-off in the process and will have to go back and update the process flowchart. Now, before you go into production, is the time to find those omissions.

The first step to testing a process flow is to "walk the process". Start by printing each process step on a separate piece of paper and then laying them out on the floor of a large room. Give yourself enough space to step from one piece of paper to another and to have other people join you at select steps. Now role-play the process: physically move from step to step, from piece of paper to piece of paper, while stating what is occurring at each point. Where the process calls for a hand-off, such as delivering a contract from Sales to the Order Entry team, hand a piece of paper representing the contract from a person on the Sales step to a person standing on the Order Entry step.

Yes, it sounds very silly but it works. By physically moving from process step to process step you activate a different part of your brain and will find yourself uncovering problems and oversights that you likely would not have discovered if you had been staring at a flowchart on a computer screen.

As you find the gaps or points of confusion within the flowchart, update the process to reflect the corrections and re-walk the steps.

## **Failure Modes & Effects Chart**

The failure modes and effects analysis matrix comes from the assembly line and industrial quality control. Here users need to make sure that any possible failure was thought through in advance and

there was a corresponding action that could be taken to resolve the problem. You can use the exact same technique for your business intelligence support processes.

**Failure Modes & Effects Chart**

Process Step	Potential Failure	How Bad?	How Likely?	Score	How Fixed?	Who Owns?
New customer order received	Order is received with less than 24 to fulfill	10	5	50	Customer launch date is delayed	Sales
New customer order received	Order is received with missing information	5	3	15	Sales contacts customer for missing info	Sales
Provision GoodData datamart	Project partially provisions (missing data from ETL)	6	2	12	Re-provision the project	Provisioning team
Set up ETL	ETL doesn't filter out non-customer data (co-mingles data)	10	1	10	Adjust ETL to properly filter data	Data Team

### The RACI chart

The RACI chart shows who is responsible, accountable, consulted, informed at any step in a process. This might not seem important, but in fact, is critical. Often when problems occur within processes valuable time is lost as people try to determine who's really in charge.

Preparing the RACI chart is simple: list each step in the process you are reviewing then, for each step, identify the person responsible, accountable, consulted, and informed about the actions being taken. Having trouble figuring out who is responsible for a process step? That's the point! Figuring out the parties involved in process steps now, when you've got the time, is far preferable to trying to figure it out in a crisis situation.



Here is an example of a RACI chart:

	Role					
Step	Sales Rep	Order Entry	Account Manager	Help Desk	Executive Management	GoodData
Receive request	Responsible	Informed	Informed	Informed	Accountable	Consulted
Implement request	Accountable	Responsible	Consulted	Informed	Informed	Informed
Perform training	Accountable	Consulted	Responsible	Informed	Informed	Consulted
Handle level 1 issues	Informed	Consulted	Accountable	Responsible	Informed	–

## Readiness planning

When bringing a product to market — whether customer-facing or for use inside your organization — it is critical that you make sure that all of the key stakeholder organizations are ready to support the new capabilities. Nothing is worse than successfully implementing a new project only to find out that you can't bill for it, the sales team doesn't know how to sell it, and it violates existing contract language.

The four areas which are vital for you to address are: sales readiness, legal readiness, finance readiness, and marketing readiness.

### **Legal readiness**

The legal readiness of the product is an area you must be absolutely sure you've covered. Of all the aspects of product readiness, this can cause the greatest delay if overlooked or damage if ignored. The best way to ensure that the legal elements of the product are addressed is by starting early. In fact, it's a great idea to involve the Legal team in the very first product workshop where you

define the goals and objectives of the project. Why? Because in order to fully understand the implications of the new analytical capabilities, the Legal team will need to understand as much about the project as the product team itself. From users to training methods to data storage and benchmarking, they'll need a comprehensive understanding of the product and you can either involve them from the outset or spend time in meetings explaining everything over again.

**Expert Tip:**

Get legal involved early in order to ensure they are on board and don't stall the process later on.

A key element to address in legal readiness is data usage. Whether you have customer personal data, credit card information, health-related data or transaction history information, the legal team will need to understand the data you plan to store, transmit, and process. If you are utilizing sensitive information, they'll be able to give advice on how to mitigate risk so that you don't open your company to potential legal issues should a problem occur.

The second element to address is service level agreement alignment. Your contracts probably offer a minimum system uptime and issue response/resolution time guarantees to your customers, but what if these don't match what your analytics provider is offering? In many cases contracts will be modified to promise "availability for core system functionality excluding our non-transactional analytical capabilities" or something to this effect, but the legal team need to be involved to determining the remedy.

Third, if you plan on offering benchmarking, work with you legal team to understand the requirement for using data in this manner. In a typical benchmarking scenario, each customer analytics instance will have access to their own data, plus access to anonymized data from other customers.

This raises several concerns such as:

- ▶ Have all the customers agreed to allow their data to be anonymized and used in this manner?
- ▶ Are the methods you've used to anonymize the data sufficient?
- ▶ Is benchmarking an opt-in only program, or can a customer opt-out of benchmarking?

Finally, you will likely need to work with your legal team to update both the contracts with existing customers and contracts you will be using with future customers. In addition to creating or updating contract sections outlining the data use policies, SLAs, and benchmarking detailed above, you will also want to inform your customers that their data will be processed by a third party (the BI vendor).

The legal aspect of the project might not be the most fun (for most of us), but they are essential to creating a successful product. Make sure you address these issues early and thoroughly.

### **Finance readiness**

Similar to the legal readiness step, you will want to make sure your Finance team has prepared as well.

The finance functions that project leaders frequently fail to address are usage metering and billing functionality.

Are you prepared to manage the number of users accessing the system, the quantity of data stored, the hours of professional services each customer has consumed? These items are frequently elements for which you will charge a customer — but can you track them? Make sure that you work with the finance team to set up processes to track all areas which determine the fees your customers

pay. If you charge based on the volume of data a customer stores — make sure that Finance has a report that shows the data storage by customer account. If you charge extra for extra connections to data sources, again, Finance needs to know.

Of course, it's not enough just to know what services each customer is consuming. You also need to be able to bill the customer for those services. Check with Finance to make sure whatever billing system they employ can handle the addition of the analytics product. In some cases they may need to create a new charge category or line item on a customer's invoice to show the additional fees.

### **Marketing readiness**

Marketing readiness is primarily concerned with four areas:

- ▶ Ensure dashboards comply with style guides
- ▶ Create/update logos
- ▶ Develop product collateral
- ▶ Website

If you are embedding your analytics within an existing application, you will want to work with Marketing early in the process to ensure that you understand and comply with the brand style guidelines. This means that will want to use the same typefaces, color palette, button types, etc. as used in the rest of the application. If the overall application uses the Helvetic typeface and the analytics are in Comic Sans, well, it's not ideal.

You might also want to work with Marketing to update existing logos or develop new logos for the analytics. This is especially true when you have unique artwork for different elements of your application (e.g., an icon for billing, an icon for new orders). If you use icons or other artwork with your application or advertising, you will want to



create a similar logo for the analytics section of the product.

Both the style guidelines and the new artwork come together in updated collateral. Analytics will be a key element of your product and you will want to update sales brochures, presentations, websites, email signatures — any place where you discuss your product capabilities will need to be revised to account for the new functionality.

The final aspect of Marketing readiness is the corporate (or product) website. Screen captures, text/copy, and demos may all need to be revisited to highlight the analytics. This can be a lengthy process so it's essential to account for this and to start early in the project.

### **Sales readiness**

You can build the best analytical product in the world and, if your Sales team can't sell it effectively, you have a major problem. Sales readiness is primarily about two areas: training and selling materials. The best practice for using training to prepare the Sales force for the new product is to educate the team in three stages:

- ▶ Initial awareness training
- ▶ Feature/functionality training
- ▶ Process training

Initial awareness training is conducted early on in the project lifecycle. The purpose of this training is not to teach the Sales team how to use the analytics, but to help them understand the capabilities of the analytical platform. Here is where you'll discuss the types of users that will be served, how they will access the analytics, and broad categories of things they'll be able to do or not do (e.g., users can drill down, users can create new filters, users cannot create new metrics). During this session you will also want to cover the estimated timeline for the project so that Sales will know if

the functionality will be available next month or next quarter.

Feature & functionality training is what most people think of when they hear “training class”. In these sessions you will show a demo of the analytics, the types of metrics, charts, and dashboards available, and the basics of how to use the system. The Sales people don’t have to be experts when the class completes, but they will need an understanding of how easy the system is to use and be fluent enough to give a demo to a potential customer.

Process training is usually conducted just prior to launch day and covers items such as how to order analytics, service level agreements and changes, customization of dashboards, metrics and charts, changes to the contract, and the support processes.

You must also consider selling materials when preparing the Sales team for the product launch. Do they have presentations showing the new analytical functionality? Are the pricing sheets updated? Do they have the latest contract template with the added analytics language? Do they have access to a demo application including logins, demo data, and a script that will help them best demo the new functionality? These are all key elements of Sales readiness.

# 10

## Launch Planning

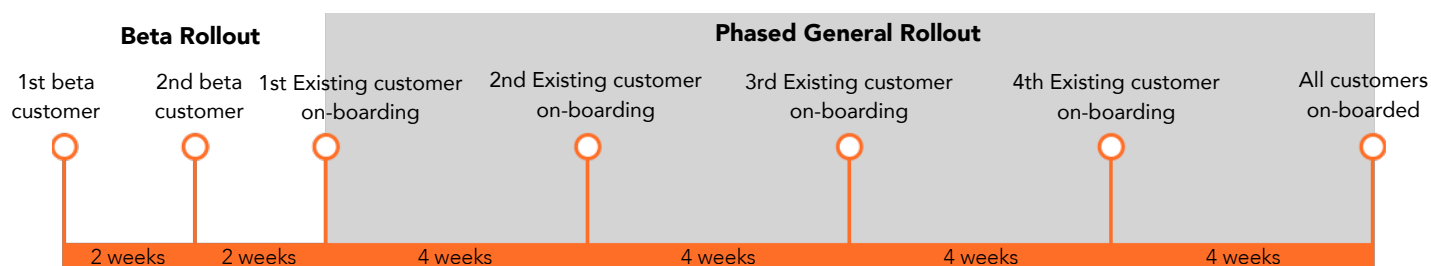
You are now in the homestretch...

Launch day is in sight and everything is ready to go. Well, not everything. You still need to plan out the details of exactly how you will be launching the new analytical functionality in your product. Which customers get the features first, how will you get feedback on issues, who decides if the launch is a success — these are the elements of Launch Planning.

### Build the rollout strategy

The first step in launch planning is to develop the rollout strategy. Your strategy should include when key activities like beta customer rollouts, general rollouts, feedback/correction periods, etc. will be taking place. It sounds simple and it is, but it's also overlooked quite often.

Begin by developing your on-boarding strategy: do you plan to roll the product out to all customers at the same time or will you stage or stagger the rollout? Staging the rollout is the preferred launch strategy — it gives you time to execute a launch, get feedback, and make corrections before the next set of customers are on-boarded. Below is an example of a typical customer on-boarding timeline:



The timeline is generally divided into two parts: the beta customer rollout and the phased general rollout.

### **The beta rollout**

During the beta portion of the timeline, you are testing not only the performance of the analytical product, but the processes required to support it. Specifically:

- ▶ **The on-boarding process** — were you able to provision the customer analytics environment, implement single sign-on, and issue login credentials (if necessary) without problems?
- ▶ **The loading of data for the new customer** — were you able to load the customer's data without issues?
- ▶ **The training or education process** — did the customer understand how to use the new features?
- ▶ **The contract negotiation** — was the new language (if any) in the contract a problem or did the customer accept it?

Of course, you are also learning about the functionality of the analytical product as well:

- ▶ Do the charts and metrics answer the key needs for the intended personas?
- ▶ Are the analytics arranged logically for the users' workflows?
- ▶ Can the user drill down to the levels required?
- ▶ Do the users have the right dimensions and filters so that they can view the data the way they require?
- ▶ Are the metrics and dashboard views self-explanatory or do the users have questions about what is being shown?

Give yourself some time between the first beta customer rollout and the second so that you can make functionality and process



corrections if needed.

It's also important that you pick the right customers to participate in the early releases. A good way to pick beta customers is to find edge case customers: those that will push the boundaries of what you might expect from a "normal" customer. For example, you might choose a customer that has very high analytical needs, that requires many metrics to run their operations as well as a customer who hasn't used analytics or reporting much in the past. By selecting both a high-needs and low-needs use case, you are likely to learn more than if you just choose an average customer for your beta.

You should plan on keeping the number of beta customers low, anywhere from one to three customers for the first beta launch. This will give you more time to spend with each customer and will also allow you more resource availability should issues arise.

### **Quick Tip:**

A phased rollout gives your team the opportunity to gather and implement product feedback from customers.

### **The phased general rollout**

Once you've deployed to your beta customers and ensured that both the product and the processes work, the next step is a phased general rollout. Now you will be deploying at scale, to large numbers of users rather than the limited release you performed during the beta, so you will want to do this launch in stages. A good strategy is to divide the remaining customer base into four groups. Grouping customers by region, by product line, or randomly all work well.

What you want to avoid is grouping customers by:

- ▶ **Size/volume:** you don't want to be launching to lots of large customers at once
- ▶ **Revenue generated:** never put all of your high dollar value customers in the same release group



- ▶ **Maturity:** having an entire group of new customers who are unfamiliar with your product in general is a recipe for a support disaster

Once you've grouped the customers into launch stages, you'll need to contact them to let them know what to expect. Having the customer account representative contact each customer followed up with an email explaining the process is a best practice. When you contact the customer, let them know:

- ▶ The timeline
- ▶ Any expected downtime
- ▶ The contract implications (do they have to sign a new contract, does the pricing change, etc.)
- ▶ The new functionality
- ▶ How they will access the new analytics
- ▶ What to do if there are problems
- ▶ How to contact you if they don't want to launch on the selected date (e.g. they have a board meeting that week and can't be down during that period)

You will want to follow-up again right before each customer group is set to be on-boarded.

### **Setting Launch Metrics & Tripwires**

How will you know if your product launch was a success? One way to measure launch performance is through the use of launch metrics and tripwires.

A launch metric (also called a key performance indicator) is a measurement of how well critical launch elements are performing.

Examples of launch metrics include:

- ▶ Time to set up a new customer
- ▶ Count of customers on-boarded to date
- ▶ Percentage of customers on-boarded vs. expected customers on-boarded (gap)
- ▶ Count of customer issues (calls, emails, etc.)
- ▶ Customer usage (logins, minutes on analytics, abandonment rate, etc.)

Track these metrics daily and make sure that all internal stakeholders have access to this information.

Tripwires work hand-in-hand with launch metrics and are indicators that something is amiss. To set up tripwires, pick launch metrics that measure key aspects of the launch process, establish acceptable boundary ranges, and then determine the actions to be performed if the metric is outside a boundary. For example:

Metric	Boundary	Action
Time to set up a new customer with analytics	Maximum of 1 hour	If the 1 hour boundary is exceeded, we will increase bandwidth to speed data load times.  We will use a contractor to help with provisioning.
Percentage of customers on-boarded vs. Expected	Maximum gap of 10%	If we have a greater than 10% gap between installed and expected, we will reduce the size of each on-boarding group and add a 5

Tripwires are critical to success — they tell you when something has gone badly enough that the team needs to take action. It's important that you establish tripwires before the launch process starts so that you don't fall into the "well, I guess this isn't too bad" trap that often catch product teams.



## Mission control

Finally, you need a mission control center for your product launch. Picture the Apollo space launches of the 1960's. As the mission commenced, you'd see the launch commander surveying the room to see if each component was ready to begin: "Main controls?" "Go flight!" "Main thrusters?" "We're a go!" and so on. If any one of those critical mission components had responded with "Well, I've got this little red light..." the whole process would have come to a halt until the issue was resolved. And so it should be with your launch process.

When preparing to launch your new product, pick a "launch control" team composed of leaders from:

- ▶ Product management
- ▶ Sales
- ▶ Marketing
- ▶ Support
- ▶ Operations
- ▶ Finance
- ▶ Legal

You should also have a launch leader responsible for running the entire process. This person is usually the product manager for the analytics, but could also be someone from the deployment or operations team.

At various stages along the launch process (such as prior to the first beta, second beta, first phase of general customers, etc.) the launch leader should survey the launch team to make sure that each critical process is ready to go. If any group feels that they are not ready and that the launch may be at risk, it is the launch leader's responsibility to delay the launch until the issues are corrected.



It is also the launch leader's job to make sure all voices are heard — you should never look back on a failed launch only to find out that someone was afraid to surface a critical issue. It's a hard proposition to call off a launch due to a potential problem, but it's better than finding the issue once it's impacted customers.

The upside to being the launch leader is just as great as responsibility. Once everything is ready, once all the issues are resolved, the teams are trained, the logos are developed, the pricing is set... The launch leader is the one who gets to say the words:

**“Launch the product!”**

**“Advanced Analytics is the number one reason our customers upgrade!”**

- **Sam Boonin** Vice-President Product, Zendesk



# About GoodData

**GoodData helps companies innovate and evolve rapidly to meet changing business demands, with trustworthy business intelligence delivered in real-time via the GoodData cloud-based platform.**

Powered By GoodData is a partner program that helps you — software vendors, cloud service providers, media agencies and system integrators — create new revenue streams from data, while delivering the best analytics to your customers.

GoodData's Open Analytics platform provides the ultimate in flexibility, allowing you to design a customized data experience, offered as an embedded solution within your own product, or as a standalone, branded portal for your customers.

Our end-to-end solution, 100% cloud-based model, and deep expertise in business intelligence give you the tools and resources you need to deliver a seamless experience with rapid time to value.

## Want to know more about GoodData?

[Contact Us](#)

[www.gooddata.com](http://www.gooddata.com)

