

Our site uses [cookies](#) for analytics. If you're unsure about it, take a look at our [privacy policy](#).



[ESPAÑOL](#) [PORTUGUÊS](#) [DEUTSCH](#) [中文](#)

Search *thoughtworks.com*



[NEWS](#)

[CONTACT US](#)

ThoughtWorks®

[Clients](#) [Services](#) [Products](#) **[Insights](#)** [Events](#) [About us](#) [Careers](#)

Experience Design [Cloud](#) [IoT](#) [Security](#) [Transformation](#) [Retail](#) [Career Hacks](#) [Financial Services](#)

| [All Topics \(26\)](#)

18 OCT 2014

How to Implement Hypothesis-Driven Development



Barry O'Reilly

Principal - ThoughtWorks, Co-author - Lean Enterprise

[Experience Design »](#)

[Transformation »](#)

[Approaches »](#)

[Strategy »](#)



7



Remember back to the time when we were in high school science class. Our teachers had a framework for helping us learn – an experimental approach based on the best available evidence at hand. We were asked to make observations about the world around us, then attempt to form an explanation or hypothesis to explain what we had observed. We then tested this hypothesis by predicting an outcome based on our theory that would be achieved in a controlled experiment – if the outcome was achieved, we had proven our theory to be correct.

We could then apply this learning to inform and test other hypotheses by constructing more sophisticated experiments, and tuning, evolving or abandoning any hypothesis as we made further observations from the results we achieved.

Experimentation is the foundation of the scientific method, which is a systematic means of exploring the world around us. Although some experiments take place in laboratories, it is possible to perform an experiment anywhere, at any time, even in software development.

Practicing Hypothesis-Driven Development is thinking about the development of new ideas, products and services – even organizational change – as a series of experiments to determine whether an expected outcome will be achieved. The process is iterated upon until a desirable outcome is obtained or the idea is determined to be not viable.

We need to change our mindset to view our proposed solution to a problem statement as a hypothesis, especially in new product or service development – the market we are targeting, how a business model will work, how code will execute and even how the customer will use it.

We do not do projects anymore, only experiments. Customer discovery and Lean Startup strategies are designed to test assumptions about customers. Quality Assurance is testing system behavior against defined specifications. The experimental principle also applies in Test-Driven Development – we write the test first, then use the test to validate that our code is correct, and succeed if the code passes the test. Ultimately, product or service development is a process to test a hypothesis about system behaviour in the environment or market it is developed for.

The key outcome of an experimental approach is measurable evidence and learning.

Learning is the information we have gained from conducting the experiment. Did what we expect to occur actually happen? If not, what did and how does that inform what we should do next?

In order to learn we need use the scientific method for investigating phenomena, acquiring new knowledge, and correcting and integrating previous knowledge back into our thinking.

As the software development industry continues to mature, we now have an opportunity to leverage improved capabilities such as Continuous Design and Delivery to maximize our potential to learn quickly what works and what does not. By taking an experimental approach to information discovery, we can more rapidly test our solutions against the problems we have identified in the products or services we are attempting to build. With the goal to optimize our effectiveness of solving the right problems, over simply becoming a feature factory by continually building solutions.

The steps of the scientific method are to:

- Make observations
- Formulate a hypothesis
- Design an experiment to test the hypothesis
- State the indicators to evaluate if the experiment has succeeded
- Conduct the experiment
- Evaluate the results of the experiment
- Accept or reject the hypothesis
- If necessary, make and test a new hypothesis

Using an experimentation approach to software development

We need to challenge the concept of having fixed requirements for a product or service. Requirements are valuable when teams execute a well known or understood phase of an initiative, and can leverage well understood practices to achieve the outcome. However, when you are in an exploratory, complex and uncertain phase you need hypotheses.

Handing teams a set of business requirements reinforces an order-taking approach and mindset that is flawed.

Business does the thinking and 'knows' what is right. The purpose of the development team is to implement what they are told. But when operating in an area of uncertainty and complexity, all the members of the development team should be encouraged to think and share insights on the problem and potential solutions. A team simply taking orders from a business owner is not utilizing the full potential, experience and competency that a cross-functional multi-disciplined team offers.

Framing Hypotheses

The traditional user story framework is focused on capturing requirements for what we want to build and for whom, to enable the user to receive a specific benefit from the system.

As A.... <role>

I Want... <goal/desire>

So That... <receive benefit>

Behaviour Driven Development (BDD) and Feature Injection aims to improve the original framework by supporting communication and collaboration between developers, tester and non-technical participants in a software project.

In Order To... <receive benefit>


As A... <role>

I Want... <goal/desire>

When viewing work as an experiment, the traditional story framework is insufficient. As in our high school science experiment, we need to define the steps we will take to achieve the desired outcome. We then need to state the specific indicators (or signals) we expect to observe that provide evidence that our hypothesis is valid. These need to be stated before conducting the test to reduce biased interpretations of the results.

If we observe signals that indicate our hypothesis is correct, we can be more confident that we are on the right path and can alter the user story framework to reflect this.

Therefore, a user story structure to support Hypothesis-Driven Development would be;



Hypothesis-Driven Development

We believe that _____ *<this capability>*

_____ *<this outcome>*

Will Result in <this outcome>

We Will Know We Have Succeeded When
 <we see a measurable signal>

We believe *<this capability>*

What functionality we will develop to test our hypothesis? By defining a 'test' capability of the product or service that we are attempting to build, we identify the functionality and hypothesis we want to test.

Will result in *<this outcome>*

What is the expected outcome of our experiment? What is the specific result we expect to achieve by building the 'test' capability?

We will know we have succeeded when *<we see a measurable signal>*

What signals will indicate that the capability we have built is effective? What key metrics (qualitative or quantitative) we will measure to provide evidence that our experiment has succeeded and give us enough confidence to move to the next stage.

The threshold you use for statistical significance will depend on your understanding of the business and context you are operating within. Not every company has the user sample size of Amazon or Google to run statistically significant experiments in a short period of time. Limits and controls need to be defined by your organization to determine acceptable evidence thresholds that will allow the team to advance to the next step.

For example if you are building a rocket ship you may want your experiments to have a high threshold for statistical significance. If you are deciding between two different flows intended to help increase user sign up you may be happy to tolerate a lower significance threshold.

The final step is to clearly and visibly state any assumptions made about our hypothesis, to create a feedback loop for the team to provide further input, debate and understanding of the circumstance under which we are performing the test. Are they valid and make sense from a technical and business perspective?

Hypotheses when aligned to your MVP can provide a testing mechanism for your product or service vision. They can test the most uncertain areas of your product or service, in order to gain information and improve confidence.

Examples of Hypothesis-Driven Development user stories are;

Business Story

We Believe That increasing the size of hotel images on the booking page

Will Result In improved customer engagement and conversion

We Will Know We Have Succeeded When we see a 5% increase in customers who review hotel images who then proceed to book in 48 hours.

It is imperative to have effective monitoring and evaluation tools in place when using an experimental approach to

software development in order to measure the impact of our efforts and provide a feedback loop to the team. Otherwise we are essentially blind to the outcomes of our efforts.

In agile software development we define working software as the primary measure of progress.

By combining Continuous Delivery and Hypothesis-Driven Development we can now define working software and validated learning as the primary measures of progress.

Ideally we should not say we are done until we have measured the value of what is being delivered – in other words, gathered data to validate our hypothesis.

Examples of how to gather data is performing A/B Testing to test a hypothesis and measure to change in customer behaviour. Alternative testings options can be customer surveys, paper prototypes, user and/or guerrilla testing.

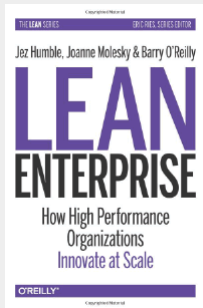
One example of a company we have worked with that uses Hypothesis-Driven Development is lastminute.com. The team formulated a hypothesis that customers are only willing to pay a max price for a hotel based on the time of day they book. Tom Klein, CEO and President of Sabre Holdings shared [the story](#) of how they improved conversion by 400% within a week.

Conclusion

Combining practices such as Hypothesis-Driven Development and Continuous Delivery accelerates experimentation and amplifies validated learning. This gives us the opportunity to accelerate the rate at which we innovate while relentlessly reducing cost, leaving our competitors in the dust. Ideally we can achieve the ideal of one piece flow: atomic changes that enable us to identify causal relationships between the changes we make to our products and services and their impact on key metrics

services, and their impact on key metrics.

As Kent Beck said, “Test-Driven Development is a great excuse to think about the problem before you think about the solution”. Hypothesis-Driven Development is a great opportunity to test what you think the problem is, before you work on the solution.



Lean Enterprise: How High Performance Organizations Innovate At Scale by Jez Humble, Joanne Molesky and Barry O'Reilly

This practical guide presents Lean and Agile principles and patterns to help you move fast at scale—and demonstrates why and how to apply these methodologies throughout your organization, rather than with just one department or team.

Download a free sample chapter.



 Recommend 4 Share

Sort by Newest ▾



Join the discussion...

**Вячеслав Гладышев** • 6 months ago

Nice article! I'd love to hear your thoughts about following issue: when we talking about experiments it usually - let's do something small quick and not very expensive in terms of resources. It is ok for a startup but what about a mature big company with millions of users? For example - there is a e-store without shopping cart and there is a hypotheses that it can be good for the store to have it. So what experiment can be done? implement whole shopping cart seems a bit overhead. implement some partial solution will risk the company reputation....

  • Reply • Share ▾**timothy_fitz** • a year ago

One aspect I don't think gets enough attention is what to do when you finally have the result of your hypothesis. A common pattern I've seen is, even with great hypotheses, when the results come back neutral or only slightly positive the business keeps the feature or change as if it had been the wild success it predicted anyway. It's rationalized with "well, anecdotally it's causing the change in behavior we wanted, but it's not showing up in the funnel. We'll figure out that part later but it's still a good change." That might be true (it's probably not), but it's frustrating to no end when you're doing frequent testing but the outcomes of the tests are not actually impacting decision making! Why bother!

Here's my preferred approach. Before you run the test, as part of writing down the hypothesis and prediction, write down a tiny decision tree of what concrete actions will be taken if the hypothesis is proven true or false. Like "if true, we'll keep the feature, if false we'll delete the feature and work on an unrelated hypothesis." Obviously you don't have to stick to it if the result is surprising in some way, or something changes, or you fall in love with the feature. But you're acknowledging that you changed from your original decision. You're forced to make better bets because if both branches are "we'll keep the feature" you know you're doing it wrong. I've also seen a great outcome where the failed-hypothesis outcome is stated up front as "we really believe in this feature, we'll try again with a different pass at it." You know you're bought into more than one trial and it doesn't feel like ignoring the outcome so much as persevering through risky bets.

  • Reply • Share ▾**Gavin van der Merwe** • a year ago

Excuse my ignorance. Isn't this just a rehash of Behaviour Driven Design? You think about the problem, model a behaviour and then empirically prove it works by way of tests. Your behaviour is in effect your hypothesis and when it passes it becomes theory. I like to

empirically prove it works by way of tests. Your behaviour is in effect your hypothesis and when it passes it becomes theory. I like the idea of lifting up measurements to your specification but like you said in this article, this approach is for uncertain times while new idea's are gaining traction. Also what guarantee's does one have that they are measuring the right thing? After all what is the universal measurement for customer satisfaction or success? Is there any more reading you would recommend that could help?

^ | v · Reply · Share ›



Barry O'Reilly → Gavin van der Merwe · a year ago

In terms of measurement I recommend 'How To Measure Anything' by Douglas Hubbard.

Too often I hear people making reasons as they why that can't try to create a measure because its too hard to measure. That's not true and simply lazy.

Your hypothesis is your target state, if you cannot describe it and create a way to understand if you are moving toward or away from it you are not actually running a valid experiment. You are simply performing experiment theatre.

There is never a guarantee you are measuring the right thing, but to measure nothing is not good enough.

As noted in Chapter 5 of Lean Enterprise: "a good technique for deciding on a given measure: "If you can define the outcome you really want, give examples of it, and identify how those consequences are observable, then you can design measurements that will measure the outcomes that matter. The problem is that, if anything, managers were simply measuring what seemed simplest to measure (i.e., just what they currently knew how to measure), not what mattered most.

^ | v · Reply · Share ›



Joseph Flahiff · 2 years ago

Hey Barry, I just got of the phone with Elizabeth, whom you met at the Lean Kata conference recently. She told me about this article. Looks like we had basically the same thoughts happening. cool to see validation from different minds. Check out how close we are in the answer to this issue!

<http://www.infoq.com/articles/...>

Copied from my article on [InfoQ.com](#)

"We (or I) believe that __ (Customer) __

would like __ (Feature or function) __

So that __ (Value) __

And to validate this we will (hypothesis test) "

I hadn't read Jeff's article but I will now.

Peace

Joseph Flahiff

www.whitewaterprojects.com

^ | v · Reply · Share ›



Sebastian Radics · 2 years ago

Thanks for the great explanation. It was fast to read and understand and with the provided template I can start right away using it.

^ | v · Reply · Share ›



Stephen Durbin · 2 years ago

Good read. I've recently been working with this train of thought in hypothesis-based consulting engagements. Very interesting to see this approach applied to agile development and this article explains some of the key concepts well.

^ | v · Reply · Share ›



Boop · 2 years ago

FYI your image is hosted on your stg server:



^ | v · Reply · Share ›



JustAnObservation · 2 years ago

Agreed Barry. The one thing I would reiterate is that TDD and BDD do not automatically give you sufficient experimental insight, just, as you say, it gives you a hypothesis statement (indeed, you could argue if there isn't data or any theory driving it, it's just a conjecture).

To conduct a sufficient enough A/B-test, you need to test against a control (at least, in cross-sectional, non-iconic models you do) and TDD also doesn't give you that explicitly. Indeed, it is that lack of analytical task in the feedback loop that can cause software developed using TDD to fall into a local optima, including getting bogged down in anti-patterns and sometimes never improve.

It would be useful to state that the level of statistical significance should be higher for higher value features (which if you consider Amazon's model, would need to be very high). This attachment of value (and hence statistical sensitivity), then makes the case for higher significance in higher impact projects much clearer. Small startups for example, only need attach a moderate level of significance to their work (relative to Amazon or [Lastminute.com](https://lastminute.com)) because the impact an outlier would have (there would be a wider range hence higher chance) would be relatively small (of course, as long as the founders haven't bet the mortgage on it).

Otherwise, good blog post. Glad to see other folk finally picking this stuff up.

^ | v · Reply · Share ›



Ken McCormack → JustAnObservation · a year ago

Great article Barry O'Reilly

>> Isn't this just a rehash of Behaviour Driven Design?

Gavin van der Merwe, I don't think so, it's a different (but linked) idea. BDD can be thought of as proving an *internal* hypothesis, i.e. that your implementation meets its *external* specification. Hypothesis-driven development is about discovering the external specification.

For example, traditionally a business stakeholder might use a heuristic like "historically, Tiffany & Co. blue has the best

results." This may still be true, but why guess, when a better approach would be to measure the performance of our best guesses?

>> using TDD ... getting bogged down in anti-patterns and sometimes never improve.

JustAnObservation, yes, although TDD is a means to implement a component in an emergent way, and primarily concerns internal code quality. They do still play a part in external specification, as unit tests are frequently driven by higher level acceptance tests, e.g. ATDD cycle, see [TDD with acceptance- and unit-test cycles](<http://www.growing-object-orientation.com/2012/05/20/tdd-with-acceptance- and-unit-test-cycles/>), everything is inter-linked!

[see more](#)

^ | v · [Reply](#) · [Share](#) ›

ALSO ON THOUGHTWORKS INSIGHTS

A Visual Journey of How Bahmni is Used

3 comments • 3 months ago•



Phil Thornley — This is a nice story. Great to hear how technology can be used for the greater good.

An Open Letter about the Australian Census

4 comments • a month ago•



marshalStacks — IMO people would be more likely to give accurate Answers IF they were not identifiable. The justifications for ID given by ...

As implicações da complexidade das camadas de tecnologia

1 comment • 7 months ago•



Miguel Gomes — Excellent article, I think you can add a layer regarding cache. Most of critical applications use cache such as memcached ...

Mudança Constante é o Novo Normal

1 comment • 7 months ago•



Rafaella Tolesani Pereira — Uau Adam! Quantos assuntos neste artigo! Eu leria novamente e encontraria mais sobre o que ...

About ThoughtWorks

We are a software company and a community of passionate, purpose-led individuals. We think disruptively to deliver technology to address our clients' toughest challenges, all while seeking to revolutionize the IT industry and create positive social change.

Connect with us



Sign up for our perspectives newsletter



[Home](#) [Clients](#) [Services](#) [Products](#) **[Insights](#)** [Events](#) [About us](#) [Careers](#) [Contact us](#)

[Press & Media](#) | [Privacy policy](#) | © 2016 ThoughtWorks, Inc.