1

Introducing Outside-in Development

"The Standish Group research shows a staggering 31.1% of projects will be canceled before they ever get completed. Further results indicate 52.7% of projects will cost 189% of their original estimates. The cost of these failures and overruns {is} just the tip of the proverbial iceberg. The lost opportunity costs are not measurable, but could easily be in the trillions of dollars."

—The Standish Group[1]

Have you ever been part of an extremely successful project? If you have, just asking this question probably has you flashing back to those days with a smile on your face. These are the projects on which members of the team really clicked. The team members grew stronger as the project progressed because they overcame challenges that at times had them wondering whether they could finish the project.

If this quintessential project delivered a software product, its success included delivering precisely what your clients wanted. You may have had a set of early customers ready to buy and become sales references the very day your product shipped. Certainly you were a member of an empowered and productive development team!

Outside-in development techniques are intended to help you re-create this success on every software product you work on. If you're like us, you want to hear applause from your clients when your product is complete, and you want to know that your hard work has paid off by positively affecting both the people who use your product and your business.

1. The Standish Group, *The CHAOS Report* (1994). The 2004 CHAOS report indicated substantial improvement, with canceled projects dropping to 18% and cost overruns down to an average of 56%. Still not a happy situation. See www.standishgroup.com.

There may be challenges to overcome

Let's look at the opposite experience.

In this case, you were probably overwhelmed with problems.

Perhaps the consumers of your products complained that the software did not work well or was hard to deal with, or that it didn't integrate with other key applications (even those your team built) or worked too slowly.[2]

Maybe these complaints were valid. In fact, maybe you'd have said the same thing if you were one of those users. So, what was the problem?

Could it be that those folks in IT operations were not providing sufficient compute horsepower for your code to run? If that was your first response, we'll challenge you to do a better job of viewing the operations team as a key stakeholder in your effort. You can help them keep you both out of this mess.

Is it that you just can't put your finger on what to change? Folks have told us, "Over the past 20 years, we've had more task forces than we can remember look at ways to deliver better software, and nothing has significantly changed. We just don't know how to do better."

Consider a different way of thinking about software product development

This book describes a way to think about software development that will increase your chances of having a successful project.

Outside-in software development is first and foremost a way of thinking about building software. It keeps the focus and energy level of the team on the people who will ultimately engage with and benefit from the product. We call these folks **stakeholders.**

Outside-in thinking asks you to be explicit about whom the stakeholders are for any of your development projects, and knowing that, to gain a clear understanding of their goals. This clarity reduces the chance of building code that won't be used or misses the point. It adds visibility to prioritization decisions and which stakeholders are affected by them. It also improves the effectiveness of conversations with the stakeholders, to maximize learning for you and the potential clients, partners, and end-users of your products.

2. We use the word *product* as synonymous with *application*. So, if you're a software vendor, think product; if you're an in-house application developer, think application or project.

You are probably familiar with multiple development processes, methods, and tools. This is *not* a book about another development process model. It *is* about ways to build software that matters to clients. It is about the thinking the team does together; how it solves problems together; how it decides what is important.

Consider some proven techniques as well

Delivering a product that allows clients to achieve their business goals demands a variety of good practices and a way to continuously improve upon them. Our practical approach to outside-in development includes many of these proven techniques.

In practice, outside-in development increases the likelihood that a product's users will be able to benefit from the code. For the development organization, this results in achieving your business goals, gaining higher customer satisfaction, and experiencing less project risk. It means better alignment of design-to-market needs, so the development team can achieve more for its time and effort. It drives toward higher-quality code which will have been built and tested in real usage contexts. It encourages excited and productive developers, because it is more fun to understand why you're building something, and more empowering to have clarity of goals. It very importantly leads to more effective deployments, because of the holistic and complete definition of stakeholders.

If these outcomes seem desirable for your development team, keep reading.

It takes a whole team to succeed

The basketball player, Bill Walton, said, "Winning is about having the whole team on the same page." This is as true when building a software product as it is for playing any team sport.

Outside-in thinking has the most positive impact when it is used throughout an organization. Success occurs when technical teams of every specialization, along with managers and executives, all understand and encourage outside-in development. So, whether you are a coder, tester, writer, product planner, team leader, manager, or executive, we expect you'll find material here that you can use.

A whole-team perspective is essential

We define development team *very* broadly: We include, of course, coders, testers, technical writers, designers, support engineers, and architects, as well as user interface designers, performance stress testers, and all sorts of other specializations.

Importantly, we also include marketers, sales executives, business strategists, product managers, business development specialists, product pricers, other finance types, and services folks as well. In our view, successful software product development is more than simply successful coding. Winning products are built by effective, cross-functional teams.

The stronger your complete team is the more fun you'll have and the more successful your product will be.

Realistically, though, some teams are stronger than others, and some teams are not strong at all. If you feel that's the situation in your development shop, don't despair. Instead, use a conversation about outside-in development techniques as a way to generate more whole-team behavior, or to help your colleagues get excited about making some changes that will improve your product. We've seen for ourselves that small successes often lead to enthusiasm for additional changes. In fact, culture change in any organization, large or small, seems to happen best when it is driven from individuals, taking their own opportunities to demonstrate leadership, and driving incremental and positive change through their teams.

Understand your stakeholders

Tensions were high. We were about to start shouting at each other.

It was what John calls an "active" meeting. The team needed to agree on the themes for a major new product release.

Today.

Laura, the product marketing lead, wanted several complex capabilities that many CIOs had specifically pointed out as important to their business.

Chris, the business development manager, wanted simpler configuration and deployment tools because business partners were getting tired of the time it took to get into production.

Don, the chief architect, thought we'd lose our technology leadership position if we didn't move to the latest engineering approach.

Bernie, the client support lead, pointed out the opportunity to lower support costs through some tracing enhancements.

This meeting was supposed to lead to a consensus view of the product content. Instead, it was a free-for-all.

What would you do?

The starting point of an outside-in development project is to understand who all of the stakeholders really are. You'll find that they constitute a varied group: some outside your firm, but some inside as well.

Stakeholders can be end-users, the clients who make purchasing decisions, the IT staffs who deploy and operate your product, and business partners who take on a range of work from outsourced operations to application development and integration.

They also can be your architects, service teams, product managers, and more.

Once you have a clear view of each of your stakeholder groups, you will need to consider their goals. Goals may vary considerably across a diverse set of stakeholders.

Only then can you determine which goals to address, and how to do so.

When in doubt, think about what will make your stakeholders heroes. You must understand the environment in which your clients operate as well as the business environments in which your software will be deployed.

Focusing on stakeholders and developing a clear understanding of their goals is the very essence of outside-in thinking. It allows you to maintain focus throughout your product's development, and it makes clear whom you must satisfy to achieve business success.

As for our example of the "active" meeting? Outside-in development techniques would have turned it from a debate of conflicting goals to a brainstorming session of which stakeholders you'd address, and what each of their goals might be. A few Venn diagrams later and you would have had a visualization of where things map well and where there is conflict.

Clarity on the whiteboard, less tension in the room.

We cover this in depth in Chapter 2, *Understanding Your Stakeholders.*

Understand organizational context

John was on a roll: six customer presentations in the past two weeks and the clients loved the new features John's team had architected for the product. Tailorable user interfaces and highly distributed device management were clearly winners.

One more briefing to go before the team closed plans for the new release. Things looked good. The briefing was about to begin.

Thirty-three minutes later, it was clear that the client had no interest in the discussion. Her body language shouted, "This doesn't interest me!"

John stopped the presentation.

"This material doesn't seem to resonate with you. That really surprises me since we've talked to other clients from firms like yours who were quite interested."

The customer seemed relieved to be asked and eager to explain:

> *"Five years ago we decentralized everything. In spite of the redundant spending, each of our 12 business units wanted the flexibility.*
>
> *But a year ago, our business climate changed dramatically. We're down to four business units and even they are running cross-marketing plays. We can't afford the current model, so we're on a big kick to centralize everything.*
>
> *In fact, what we want most is a single common interface that can't be modified, along with a centralized management model. This seems to be the opposite of what you are focused on building."*

What happened here?

Your clients are affected by the structures, cultures, and initiatives of their own organizations. Because your clients will make purchase and deployment decisions that reflect their organizational contexts, and they probably want to use your products to effect further changes, you need to understand these structures.

Once you do, you can more effectively determine what capabilities your products must deliver in order to help your clients achieve their goals.

By factoring out noise due to organizational context issues from the dialogs you have with your stakeholders, you'll get feedback more efficiently and get to important insights more rapidly.

That really would have helped John's situation.

Yes, he was able to recover the meeting by explaining that the product had centralized capabilities too. But he needed outside-in development techniques to figure out what to do with the conflicting client goals that were so closely tied to organizational directions.

This is not an isolated case. Your clients are increasingly looking to change their organizations as they seek efficiencies, flexibility, and market differentiation. They will rely on software—perhaps your products—as an enabler of these changes. You need to understand how to deal with these situations effectively for your products to be relevant.

We cover this in depth in Chapter 3, *Understanding Organizational Context.*

Make your products consumable

Have you ever bought a really well-wrapped product? Carl recently bought a memory chip for his laptop. It was very securely wrapped. So much so that he had to struggle to open the outer hard-plastic container; it seemed to defy even the sharpest of scissors. He finally had to use a knife to cut through the wrapping, but he was worried about damaging the chip inside. When he finally got the item out, he sighed with relief.

This is an example of consumability in practice.

What does this have to do with a software product?

Everything.

As the client stakeholder, Carl's goal was to use his new memory chip. But first, he needed to complete a variety of tasks that had nothing to do with his goal. He had to unwrap it, and that, as you saw, was no easy feat. Once he finally got that done, he still had to install it in his laptop, and then verify that it worked correctly. All before getting to the point of satisfying his goal: to use the new memory chip.

We call these extraneous tasks **meta-tasks.** Yes, they are often necessary. But they are often unnecessarily difficult, time-consuming, tedious, and intrusive. People who use software products encounter these meta-tasks all the time: in activities such as purchasing, installation, deployment, and integration to operations, problem management, and upgrading.

Consumability is all about minimizing your stakeholders' barrier to goal attainment.

We cover this in depth in Chapter 4, *Making Products Consumable.*

Align with your stakeholders' goals

Imagine that you and a friend are on two motor boats. Hers is moving, and it's a few thousand meters away from yours. You want to motor up to be next to her. You aim at her boat and run your engine.

Will you need to steer your boat again before you reach her?

Of course you will.

Especially because as you start out, you can't even predict which way her boat is moving.

You need to stay in alignment. You need to ensure that your progress tracks to where your friend is moving so that you can intercept her—or at least get close.

Software development has an equivalent problem. Your stakeholders will keep moving, because their perception of their business problems and how best to solve them will change. This is because their environment will also be in motion, with pressures from within their firm on different aspects of the problem, changes in technology, your competitors' product offerings, or organizational changes.

Your understanding of the stakeholders' goals needs to keep up.

Meanwhile, your team is learning things too. As you progress in product development, you discover different ways to provide solutions to the stakeholders' goals. You get a better understanding of the overall situation. The more you share your vision of how your product will operate with your stakeholders, the more useful feedback you will get. Interestingly, the more your vision is shared, the more you cause your stakeholders to also modify their thinking based on the possibilities they can now imagine.

Continuous stakeholder alignment might seem a bit overwhelming at first, but there are proven techniques to manage it for optimum results.

We cover these practices in depth in Chapter 5, *Aligning with Stakeholder Goals.*

Define success in your stakeholders' terms

Have you ever been to a "final ship" party? The product is complete, testing is over, and the download images or discs are built. Tomorrow morning the product will be available to customers. It's over.

Or is it?

Don't get us wrong, we like parties too.

Let's just be clear what it is we're celebrating.

It definitely is time to celebrate the completion of a major milestone. But it isn't success. Success comes when your clients can achieve their business goals through your product.

You are, however, now firmly in the production phase of a software product's life cycle. You're about to face three waves of work, each requiring specific outside-in development techniques to adequately manage.

The first wave is brief, and it continues until a target number of early-adopter clients declare success with the product. The second continues for quite some time, until the next milestone event of shipping a follow-on release. That signals the start of the third wave: long-term product support.

Because outside-in development is a stakeholder-based approach, it should be no surprise that success must be defined in the stakeholders' terms. Beyond that, a variety of OID techniques can assist you to succeed with your stakeholders in all three waves and to leverage your experiences to improve subsequent products.

We cover this in depth in Chapter 6, *Defining Success in Your Stakeholders' Terms.*

Become an outside-in developer

Outside-in development is a way of thinking about software development supported by a set of techniques. You might wonder, "How do I get started?" Let's first point out what you *don't* need to do:

You don't need to change your entire firm.

You don't need to change your development process.

You don't need to change the tools you use today.

You don't need to change the way everyone in your organization does his or her work.

Okay, now that we've taken some of the difficult items off the table, here is what you *can* do.

You can do OID in place; in other words, inside your current set of practices and processes. With your current tools. Within your immediate function, if you'd like. Any member of a development effort can exert tremendous positive change on the whole team.

The leaders within the development team who can drive the adoption of OID are not just those who have leadership titles or positions. A number of tips for the effective deployment and use of outside-in development will make it an easier and more enjoyable way to work.

We cover those techniques in depth in Chapter 7, *Becoming an Outside-in Developer.*

The leader's role in outside-in development

Outside-in development thinking and techniques are accessible to every member of a development team. Leaders of every sort have an additional responsibility and opportunity to help their colleagues, enable a successful development experience, and, of course, produce a winning product.

Multiple organizational and cultural challenges can occur in deploying any new style of thinking about software development. Even when your teammates understand OID and are eager to practice its techniques, they may be nervous about working closely with external stakeholders. Plus, most development teams comprise a wide range of personalities that react to change with varying levels of enthusiasm and panic!

Although there is no magic potion to fix every issue, there are techniques and tips to help you succeed. If you are a leader, you will recognize which items apply explicitly to you or require your specific focus.

Look for a section like this in every chapter; these will bring your attention to topics of importance to development leaders of every stripe.

Essential point: You can get started now

No additional preparation is required to get started. You need not even finish every chapter in this book before you use outside-in development techniques on your product.

Carl often reminds his teams, "On the way into work in the morning, no one says 'I think I'll do mediocre work today, maybe make life difficult for my clients, and not really focus on helping them benefit from my product.' "

Seem like a silly reminder to you?

It isn't.

At least not until every member of your team says, "I'm going to do great work today, make it easy for people to work with my product, and definitely focus my efforts on what it takes for my clients to be heroes!"

This kind of thinking won't happen magically.

But it *will* happen.

You just need a framework, a way of thinking about things, and some techniques.

Jump in now; there's no time to waste. You have winning products to deliver.