Starting with the End in Mind

How Solutions ITW Refactored, Rebuilt, and Reintroduced Its Flagship Software Package

Solutions
IDEAS THAT WORK

Solutions ITW
620 North Main Street, Suite 200
Greenville, SC 29601

864-404-3265
info@solutionsitw.com
solutionsitw.com/solutions-dgr

TABLE OF CONTENTS

INTRODUCTION

Since the beginning of the software development age, BETA releases and bugs have gone hand-in-hand, and for good reason. Software development is the translation of logical ideas and strategies into the digital medium. The source of this logical thinking is human beings, and as long as humans are flawed in their logic and translation of that logic to the digital medium, there will be bugs.

BETA releases represent code that, while having undergone significant testing in the lab, has not gone through significant testing "in the wild," the space we call "the real world." In this space, developers present their preciously-crafted applications (digital translations of human logic) to be used and validated by other logical beings.

Surprise! Those beings employ different logical thinking than the developer.

"I never thought of that," and "You aren't supposed to use it like that," and similar become the developer's mantra during this BETA testing period as they see bugs roll in from the real-world usage of their application. This process repeats itself over and over until the developer and the user come to a logical agreement and validation for the usage of the application. The duration of this process depends on how adept the developer is at anticipating the user's thinking and behavior.

Wouldn't it be great if logical beings could anticipate what other logical beings think? How advantageous would it be if logical beings could communicate with other logical beings to share their ideas/logic to shorten the BETA process and bring applications to General Release more quickly?

We at Solutions ITW proposed to do this in the release of our new platform, Solutions DGR 5.0. Our goal was to port our 17-year-old-.Net-Windows-code-base to a web-based platform with little-to-no downtime for our clients... and we succeeded!

BACKGROUND

To be sure, we are not the first people in history to have posited that greater collaboration between developers and users leads to better software quality and improved customer experiences. This is precisely what the 2001 Agile Manifesto (http://agilemanifesto.org/principles.html) proposed in ground-breaking fashion. We at Solutions ITW simply employed these principles to see if we could achieve the results we wanted/needed.

Solutions ITW has been helping Goodwills implement Donated Goods Retail (DGR or Thrift) management solutions to meet the challenges of Goodwill's unique retail needs since 2006. Our team of 17 employees works with our Goodwill partners to configure a uniquely implemented system that allows each Goodwill to manage their thrift operations the way they want their operations managed. And because the system walks employees through best practices and helps to enforce retail processes, Goodwills can realize strong ROIs much more quickly. Employees are successful while helping the organization succeed. Our mission is to help Goodwill succeed - at all levels, and we believe Solutions DGR accomplishes that mission.

However, the application was dated. Whenever a new version was ready for release, each workstation had to be touched at every location (600+ at the time of the release of the 5.0 Production Module update). With an average of 7 workstations at each location, keeping all clients up to date in a timely manner was untenable. Beyond that, by the time a location had been updated, the following application version release would be ready, and the client would be behind again.

Added to this was that all bug fixes required the same amount of labor to roll out. There was no light at the end of the tunnel. All our time was spent treading water (i.e., keeping the current system running), and we were failing. We had to concede that the platform needed to be more scalable and was in danger of becoming obsolete. We also concluded that, while the platform provided significant value to our clients, that value was slowly diminishing due to the effort required to roll out new features and bug fixes. The only way to continue to serve our clientele well was to refactor the Solutions DGR application and platform completely.

Our clients agreed with our assessment. By the time we arrived at this conclusion, most of our clients had their very own version of the software due to version fragmentation. Bugs lingered and promised features never arrived. Our clients let us know, in no uncertain terms, that something had to change.

So, we began the most aggressive and consequential project in our history, the complete refactoring of our flagship application.

EXPLANATION

VISION

Committing to the re-architecting and rebuilding of a company's foundational software package is not a small commitment. It is akin to rebuilding the plane while it is flying. We were keenly aware that a significant misstep at this stage could ruin the company. As such, our team committed itself to the ideal of the second habit:

Start with the end in mind.

While this was the largest project our team had ever undertaken, we had completed significant projects in the past. As a team, we learned that not knowing where we were headed was a recipe for disaster. The decision to envision not only the final product but the process to get there had the most significant impact on the success of the project. From the start, we outlined the following definition of success for the project:

1. *The application had to be easy to use with limited training.*
   Our clients did not have time to retrain their entire workforce. It could not be that in our attempt to improve the platform for our clients, we would be adding to their burden.

2. *The application could not cause downtime for clients.*
   We expected there to be bugs and issues, but those issues could not lead to our clients being unable to make money. It could not have any noticeable negative impact on their operations.

3. *The application had to have the same feature set as the existing application.*
   We had worked hard to build an unrivaled application in the industry, and this new platform needed to be a step forward. We needed to provide the same value in this platform that was already available in the existing.

4. *The application had to build confidence.*
   An application that frequently crashed, was difficult to navigate, or produced incorrect results would not inspire confidence. We needed an application that worked how our clients needed it, even in the BETA phase.

   We also needed the platform to inspire our development team. They would be the ones to maintain and expand the platform. They needed to have a platform they trusted and even enjoyed.

DESIGN

In keeping with the Agile principle of fostering differing perspectives and collaboration during the design phase, we pulled experienced employees from all departments to design the new application and platform. We gave them the vision (boundaries) for the project and asked them to answer the following question:

If you had a magic wand and were able to start over with our platform, what would the final product look like?

We then asked this same question to several of our closest clients. Based on both sets of feedback, we created the following list of objectives for the project:

- Data Transfer Stability
- Operational Stability
- Change Stability
- Standardized Environment
- Single Version
- Quick Change Release
- Self Maintenance (through automation)
- Self Implementation (through automation)
- 1-Step Deployment
- Cross Platform

We were surprised that multiple changes/wishes/requests were reported back from our clients that needed to be added to our list. Our client interaction forcibly reminded us that our clients would only consider this platform change a success if it met their needs. So we tabled some of the changes included in our internal review and prioritized our customer requests instead.

FAILURE

With the list of enhancements and requests in hand, we would like to report that the process of scoping out the work, creating tasks, creating sprints, assigning tasks, and doing the work was a remarkable success. However, that would not be the truth.

We struggled mightily as a team. We knew where we wanted to go and what we wanted the end product to look like. We had the full support of our team and clients. We were fully funded to make this leap. **But we needed more operational experience and industry knowledge to move our team**

effectively toward our goal. In deconstructing the most significant challenges during this time, we have determined that communication, in its many forms, was lacking for this project:

- **Developers did not have a firm grasp of all the work that had to be done.**

  Senior Management was vague in its communication, “Include all the functionality already in the existing platform.”

- **Senior Management did not realize that the developers were struggling.**

  Developers put their heads down to do the work assigned to them, were self-reliant in pursuing answers to their questions, and did not communicate challenges back to leadership.

- **Senior Management was not able to see progress (or the lack of progress)**

  Developers were able to show UML designs and code windows and were even able to show working code. They were not able to show any interconnected code working. But Senior Management did not pick up on this.

  Because of this, it was always a surprise when dates were missed because “we had been making such good progress.”

At this point in the project, about 1.25 years into a 1-year project, we began to fear that we had made a terrible mistake.

SUCCESS

It was only when we were able to hire a new DevOps Manager that things improved. While we enjoyed significant success in other areas under our previous DevOps Manager, our new DevOps Manager brought more real-world development management skills to the table. In a matter of months, our team was able to visually see where we stood, though we were still struggling to see what still remained to be done. Regardless, we were taking small steps forward.

Our morning stand-ups turned from complaining sessions to identifying what needed to be done and who was going to do the work and then became one of reporting the work that one had done the day before while identifying what they would be doing that day.

The communication between the developers and Senior Management improved, and we learned that the developers were missing significant swaths of information. They had a general idea of what needed to be done and the functionality to include, but they lacked the definitive list. For the first time, our senior management sat down to make a comprehensive list of the functionality required which was then divided into tasks and estimated. We began to see the light at the end of the tunnel and had real hope that the project had a chance of succeeding.

The good news is that while it was confirmed that there was miscommunication among our team members, there was enough leadership on the team to produce a solid core for the new application. We were able to progress quickly under new management and were able to see immediate results.

It is important to note that while communication struggled internally, our team did a better job of remaining in communication with our clients. We repeatedly presented them with the vision of the project (listed above) and repeatedly received validation of that vision from them. By the time we reached BETA, our team was confident that as long as the system performed as intended, our clients would be happy with the direction we had taken with the platform.

BETA

Our team felt confident that they had done enough work to begin planning to deploy the BETA for Phase 1 of the project. We identified a client who had kept close contact with us during this process and drafted a BETA contract for them to sign. We wanted our BETA testers to know that we were expecting things from them. For instance, we expected the following:

- Daily usage and testing
- Daily feedback in a specific format
- Daily progress toward general release
- Daily stand-ups
- Full and complete compliance with instructions
- A positive attitude and demeanor

If this client were unwilling to participate in this manner, that would be fine, but we would need to find someone else. We were very fortunate to have this client willing to participate in this way.

PRESENTATION OF FINDINGS

With the functionality for BETA ready to be released and a willing client in hand, again we would like to report that the next steps flowed like clockwork. While it was not as messy as previous issues, we did encounter some difficulties even in the rollout of BETA.

Our development team was unwilling to release the platform for BETA.

The reason? There were known issues in the platform that still needed to be addressed, and our developers wanted to ensure that the platform was perfect before they released it to the client for BETA.

We had some internal conversations on the matter and asked ourselves what the purpose of BETA was if not for the client to knowingly take an unfinished product and work with it. While we appreciated that our team wanted to put our best foot forward, we needed client feedback in the form of testing as quickly as possible. What we communicated was that as long as the client was able to work with the application, it was time to release the application to them to see how the platform would stand up to user testing. To be sure, we had been getting client feedback all during the process, however this was going to be the first time we would be able to give them working code.

In July of 2022, we released Phase 1 of our Solutions DGR platform refactor to BETA. With the release, we included a list of known issues and gave our testers direction on what to test. We asked them to set up one workstation that would be used for as long as possible. If the application did not crash or present any other issues, we expected the testers to use the application all day. The first day the users had the application, they were able to use it for a few hours before having to revert back to the existing version. The developers made changes to the application, and our implementation team was able to update the version the testers were to use by utilizing the installer application that we had created to simplify roll outs. The BETA testers were given the revised application later that week and have been able to run it for entire days ever since.

In the time that the application was in BETA, it was only down for half a day. During that stretch, the development team needed to reverse a bug that was introduced which caused all data collected to be lost during a 3-hour stretch of the day. Other than the initial issues and the data-loss issues, the BETA experience with the application was an unmitigated success, and the application emerged from BETA a month later, and Phase 1 of the new platform has been running in multiple live environments for six (6) months with minimal issues.

CONCLUSION

There are several factors that we believe contributed to the success of the BETA phase of our new application.

COMMUNICATION

As already mentioned, we believe that our close communication with the BETA testers before, during, and after the BETA phase contributed greatly to the success of the BETA testing phase.

Our testers knew ahead of time what to expect from the application because they were part of the process to get to where we were. We had several design meetings to lay out how the application was to work and what we wanted the user experience to be, not just in the use of the application, but also throughout the BETA process.

Our testers also knew what was expected of them because they signed the BETA testing contract and were in close contact with our team.

We also had daily stand-up calls with our BETA testers, reinforcing the principle of short feedback loops, for two reasons. First, to ensure we were fully aware of the latest user experiences, and second, to ensure that testing was taking place. We were laser-focused on exiting BETA as quickly as possible, and we needed our testers to be moving at the same speed.

All of this communication ensured that all parties knew what to expect from each other and to stay on the same page during the BETA.

BACKWARDS COMPATIBILITY

One of our key requirements throughout the entire process was backwards compatibility. Our clients had to be able to:

1. Run the new platform in the same environment and using the same workstations they were running the existing environment
2. Switch back and forth between the new platform and the old in the event of show-stopping bugs

To meet this requirement, our team decided that while we might add to the underlying database schema, we would not alter it. As long as we ensured that data from the new platform was in the

same format as the existing platform, we would be able to run both platforms on workstations that were side by side. This accomplished a few things.

1. Our clients were able to set up a workstation initially as a training station for the new platform in the same location as workstations running the existing platform.
2. The stores were able to run one workstation on the new platform while running the other workstations on the existing platform giving them a sense of control and stability.
3. If, for whatever reason, the BETA application decided to start having problems, the client was able to revert back to the existing platform on that workstation, and at the end of the day, all the data would still flow into the same database schema without any extra effort. This ensured that there was never any manipulating or massaging of data between the two platforms.

Our avoidance of an all-or-nothing approach to running our BETA phase allowed our clients a degree of confidence in rolling out this untested application in their LIVE environments. They had a clear rollback path. They also had a clear training path since they were able to demonstrate the new application side-by-side with the existing application and could train without changing their process.

EASE OF ROLLOUT

One of the most significant hurdles we were seeking to overcome with this new platform was the need to perform database updates at each store running our software platform. At the time of the release of the beta, we had the application running on over 6,000 machines. Most of these machines were on disparate versions, meaning that they were not even running the same database schemas. It was a significant challenge to create a set of database update scripts for each one of these installations.

With that in mind, our team proposed to provide for widespread application deployment. We created an installer application that communicated with our cloud environment and was able to see which versions were available for download and which version was the latest BETA and LIVE version. It displayed this information to admin users and allowed them to use the latest version or to download a specific version.

Our BETA testers were able to control which versions were on which machines using this tool. This greater level of control allowed for BETA testing to continue even if a bug were released into the latest version. Instead of having to revert back to the application's existing version, the testers could download the previous version of the new application (e.g., one without the bug) and continue testing the new application.

In addition to the added continuity provided by the installer, the testers were able to install the application on new machines in 2-3 minutes with just a handful of clicks. Once the application was installed, the user would run the application which would then download the settings for that location (i.e., the same settings in use by the rest of the location on the existing version), and within 30 seconds of starting, the workstation would be ready for use.

Last of all, the design team decided at the outset that we would keep the UI/UX experience as close to the existing experience as possible. Colors were changed. Fonts were manipulated. Some workflow aspects were modified, but the feel of the application remained essentially unchanged. So much so, that one of the BETA sites ran into a situation where they quickly needed another workstation put into service and opted to use the new platform. Users were up and running on the new workstation with less than 15 minutes of training due to the similarities in workflow.

This ease of rollout and ease of use has led to strong adoption of the new platform even during the BETA testing phase.

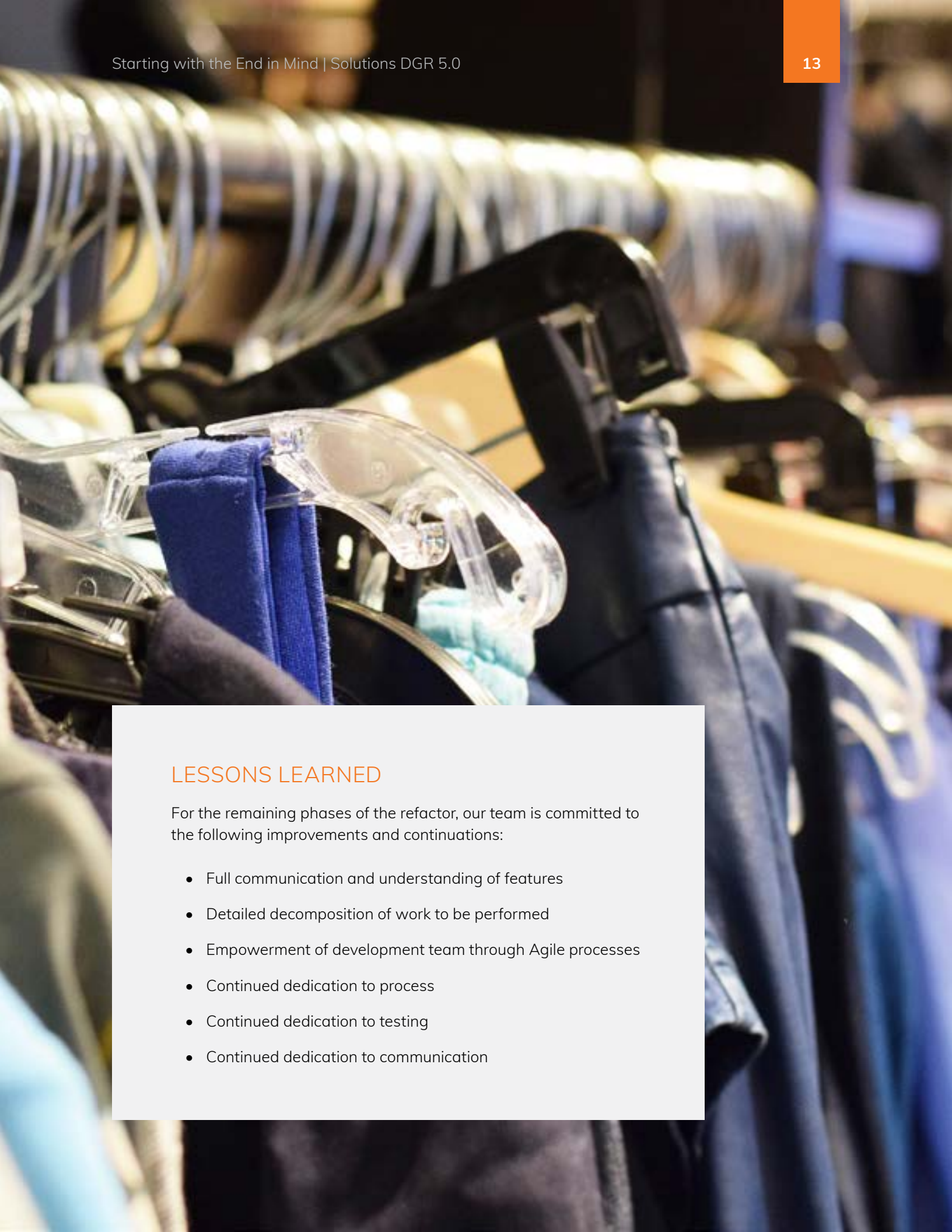
BETA EXIT

Prior to BETA testing, we performed our ALPHA testing in-house. Several of our implementation and support team members put the application through its paces until we reached the point where we "forcibly" released the platform to BETA (see above). By the time ALPHA testing was completed, our team was feeling good-ish about the platform.

The goal for our BETA testing was to thoroughly test the application and get out of BETA as quickly as possible. We set as a goal to have no new bugs reported over a seven (7) day stretch and no bugs currently in the system as our benchmark. When we hit this goal and because of the success we had already experienced in ALPHA testing, we believed that the system would be solid enough for General Release.

What we did not expect was that we would reach this mark so quickly. Specifically, we exited BETA a little over 3 weeks after we started the BETA phase!

Since that time, the application has been rolled out to approximately 109 locations with over 600 daily users. To be sure, additional issues have been identified, but to date, none of the issues has caused downtime, and all locations have been able to use the application full-time. Adoption of the application has yet to encounter much resistance if any, and the success of the BETA has instilled confidence that the new platform is stable. As a result, these locations have ceased using the previous version of the application and are running on the new platform.



LESSONS LEARNED

For the remaining phases of the refactor, our team is committed to the following improvements and continuations:

- Full communication and understanding of features
- Detailed decomposition of work to be performed
- Empowerment of development team through Agile processes
- Continued dedication to process
- Continued dedication to testing
- Continued dedication to communication

SUMMARY

Our intent was to replace our existing flagship platform with a newly-refactored platform while ensuring the application:

- Was easy to use with limited training
- Did not cause down-time for clients
- Had the same feature set as the existing application
- Built user confidence

We set out to start with the end in mind, foster an inclusive design process, and encourage short feedback loops. We encountered challenges during development:

- Developers did not have a firm grasp on all the work that had to be done
- Senior Management did not realize that the developers were struggling
- Senior Management was not able to see progress (or the lack of progress)

We overcame these challenges through improved leadership and communication and were able to successfully and quickly navigate the BETA Testing phase due to our team's dedication to:

- Internal and External Communication
- Backwards Compatibility
- Ease of Rollout

Our team's dedication to process and communication allowed our team to deliver this new platform and to meet the defined goals.

Solutions
IDEAS THAT WORK