CONTINUOUS DELIVERY in **90** days

AGILE APPROACH PREVENTS IMPLEMENTATION FROM FAILING

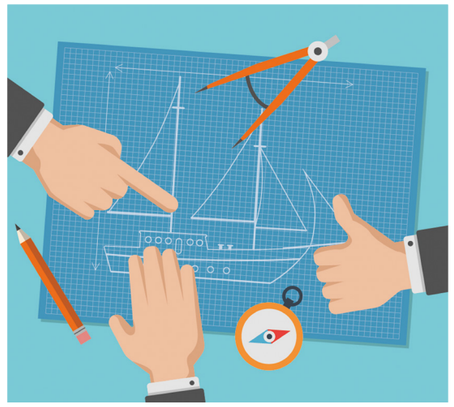
CONTINUOUS DELIVERY INCLUDES THE AUTOMATION ALL MANUAL ACTIONS AROUND TESTING, INTEGRATION AND DEPLOYEMENT OF SOFTWARE. IN PARTICULAR WITH WORKING AGILE, IT IS INEVITABLE. HOWEVER, USUALLY IT IS STARTED TOO LATE, OR ENDS UP IN DISCUSSIONS ABOUT TOOLS. AND THAT IS A SHAME, BECAUSE THERE IS AN APPROACH WHICH MAKES IT POSSIBLE TO HAVE CONTINUOUS DELIVERY IN PLACE IN ONLY NINETY DAYS.

By Robert van Vark & Rini van Solingen

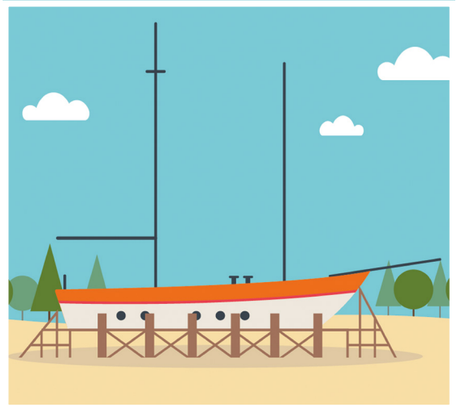
AGILE WORKING IS BASED ON ITERATIVE DEVELOPMENT. Most organizations nowadays work with iterations of two weeks or less. This has the result that software can be released more often. This asks for more extensive automation of Unit Tests, Acceptance Tests, Regression Tests, Integration- and Deployment Processes. All these forms of automation in practice are combined as Continuous Delivery (CD). In practice there is however not much attention for CD. Many organizations look up to it and view it as a gigantic project. And this is also true. Implementing CD and removing all backlogs and legacies considering automated testing, is no sinecure. But actually,

"DON'T SPEND AN ETERNITY ON SELECTING TOOLS"

it is not really a choice anymore. The amount of releases when working Agile makes it impossible to test everything manually. Besides, not testing is simply too risky. This is also the reason why most organizations who



Assessment



Implementing Blueprint

Launch Teams

This article was translated from **AG Connect** - April issue

CONTINUOUS DELIVERY IN PRACTICE: SIX TIPS

1. SELECT TOOLS AND START IMMEDIATELY

Good tools are important, but in the end, the teams that make the difference. Therefore, do not spend an eternity on selecting tools. Tool selections tend to be strongly focused on costs or on the completeness of features to be prepared for any surprise. A deep pitfall is to keep searching for the 'ideal' tools. If they even exist, and finally are found, in the end it will turn out they are not able to integrate with one another. The ideal world does not exist, so do not strive for it. Use a standard set which is commonly used in the market and start there.

2. CREATE AN IMAGE OF THE COMPLETE WORK

First, do an assessment to prevent surprises in the trajectory. Without intake assessments everything may seem to be possible, which could result in some figurative skeletons in the closet upon implementation. A proper assessment gives insight if CD is the correct next step or if other aspects might need attention first. Is the application even suitable for Test Automation?

3. LOOK BEYOND DEPLOYMENT AUTOMATION

Many organizations make the mistake of having a limited view on CD. They merely see it as Deployment Automation and nothing more. Factually, they want to automate existic manual installation actions by administrators and not go beyond that when in fact, it is about more than only deploying. It is about making an entire system work and keeping it that way. This makes CD much broader and adds integrations, Automated Testing, Resilience processes and Configuring.

4. AUTOMATE AN IMPROVED WAY OF WORK

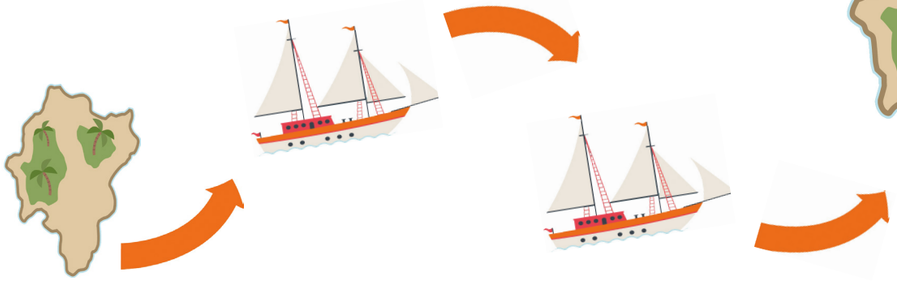
Try to not literally automate the current process. This is often created manually and with focus on the long-term. Try to work more effectively and efficiently. CD helps because simplifications are also easier and more quickly testable. A structured roll-out of Continuous Delivery hence, helps to simplify and accelerate the current IT landscape.

5. MAKE SURE EXPERTS HELP THE TEAMS ALONG

Nothing is as frustrating for a team as knowing what the goal is, but not being able to see how to get there. When a team is stuck on the same problem for days, this takes a lot of energy and it might even endanger the CD implementation. It is essential that in that case, someone is available who can immediately help the team. Such a problem might then provide energy and motivation because the team will see that even such cases can be solved. For this, access to expertise is needed.

6. SEPERATE STEPS, EACH WITH DIRECT VALUE

Try not to solve all problems at once. Yes, 'automate everything', but only everything that is already there. Many organizations go for the 'Full Monty' when implementing CD. Everything they did not do before when an application was installed in production, the want to automate fully. Think of things such as Infrastructure as code, dynamic on-demand provisioning of environments, automating acceptance tests which did not even exist beforehand. Do not do it. Start small and build on that. How do you eat an elephant? Indeed, bite by bite.



have started to implement Agile, use Continuous Delivery as their next step. Then follows the next difficulty: trajectories towards CD often quickly silt in lengthy decision-making processes about tools and technology. And each month that this choice still needs to be made and the decision is postponed, implementing CD automatically shifts to the future. Before you know it, three months will have passed doing proof-of-concepts and comparing tools. All that delay is a waste. It only puts off the solution to the problem.

Alternatives

There are alternatives to prevent this and they have been proven in practice. De essence is that implementing CD is complex and therefore works best if done in an Agile way. The best teams incrementally get rid of their backlog towards CD for their own applications. This ensureS speed and makes teams directly experience the benefits and results. This in turn saves time for taking the next step. This seems obvious, but in practice, most organizations still make the mistake of wanting to implement Continuous Delivery with a big bang and using Waterfall. The past has taught us that this usually fails.

"DESPITE THE LARGE PILE OF WORK, DELAYING MAKES NO SENSE"

Authors

RINI VAN SOLINGEN is CTO at Prowareness (rini@scrum.nl) and part-time professor at TU Delft. He is one of the lead trainers of the masterclass: www.leading-agile-transformation.com

ROBERT VAN VARK is Principal Technical Consultant at DevOn (r.vanvark@devon.nl. To take the first steps towards CD in 90 days, check out www.devon.nl/continuous-delivery-assessment.

The most important lesson therefore is: impelementing CD is most usually succesful when done iteratively. The proper approach consist of three phases:

1. **Assessment**
   The assessment maps the current way of working and present skills. At least as important is a technical assessment of the application. Should CD be the first step at this point? Or should the application be unbundled to make it more testable or easier to deploy?
2. **Implementing blueprint**
   A blueprint for a Continuous Delivery Infrastructure is used to start up the CD pipeline for applications. This is the preparation, so teams can automate tests themselves and can deploy automatically.
3. **Launch Teams**
   With help of training, coaching and 'starting immediately', teams are equipped with all knowledge and skills needed to incrementally automate everything themselves.

Three months

Practice shows that with this targeted approach, it is possible to get teams to the level where they take along CD in their iterations. Thus, they step by step are taking over the backlogs on automated testing, integration and deployment. When teams take their own responsibility, focus is usually on: direct effect since teams focus on automating the steps which have the greatest effect. This might be either in saving time, taking out boring work or it might help in improving quality. In short, Continuous Delivery is inescapable when organizations start working Agile. Besides the huge amount of work, delaying makes little sense. Starting sooner is way more effective. It is wise to use an Agile approach in which step by step, small pieces of work, are 'automated out'. Practice shows that within ninety days, teams are able to work on this individually. 'Just do it' is therefore a simple, but one of the best recommendations.

This article was translated from **AG Connect** - April issue