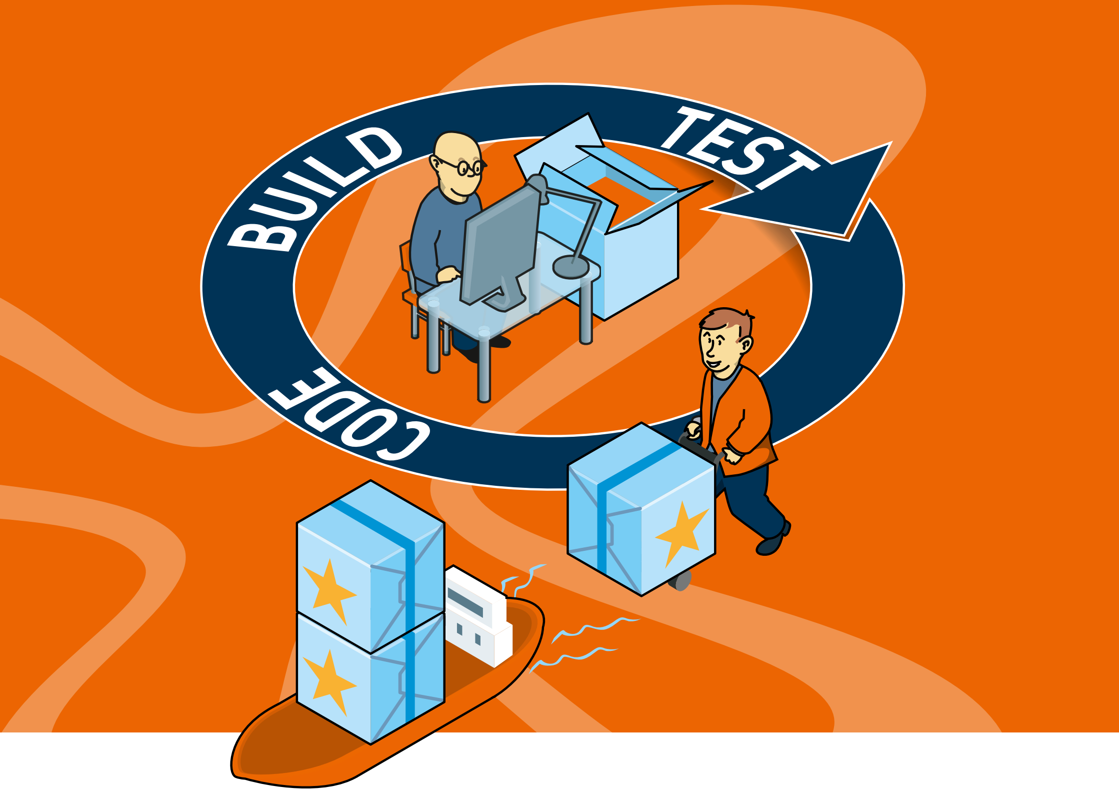
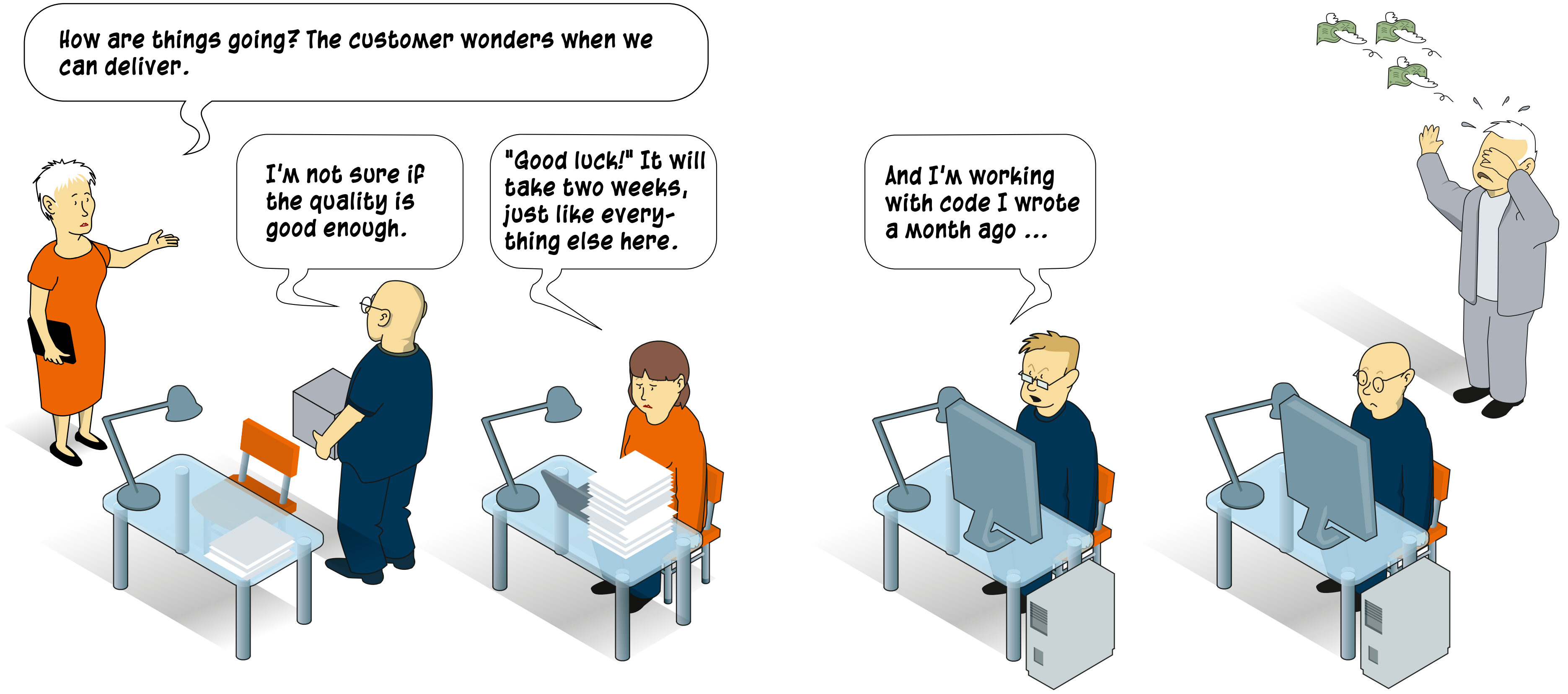
CONTINUOUS DELIVERY

in 5 minutes



SOFTHOUSE

JUST ANOTHER DAY AT THE OFFICE ...


How are things going? The customer wonders when we can deliver.
I'm not sure if the quality is good enough.
"Good luck!" It will take two weeks, just like every-thing else here.
And I'm working with code I wrote a month ago ...

Does your organization need to tighten up quality and delivery reliability – and customer satisfaction?

Then we recommend taking a closer look at Continuous Delivery – a method where:

- You always know the status of code: ready for delivery!
- The team quickly gets confirmation that everything works after changing any code
- You can give the customer something to try and feedback on at an early stage
- Stress levels drop and the space for creativity increases
- It is quick to adapt to changing conditions and customer requirements
- Your developers will not have to go back to old problems that should have been solved a long time ago
- Your organization can provide a reliable delivery time plan that customers can trust

Continuous Delivery is a method that allows developers to focus on writing good code; they know that what they deliver works together with other parts of the software.

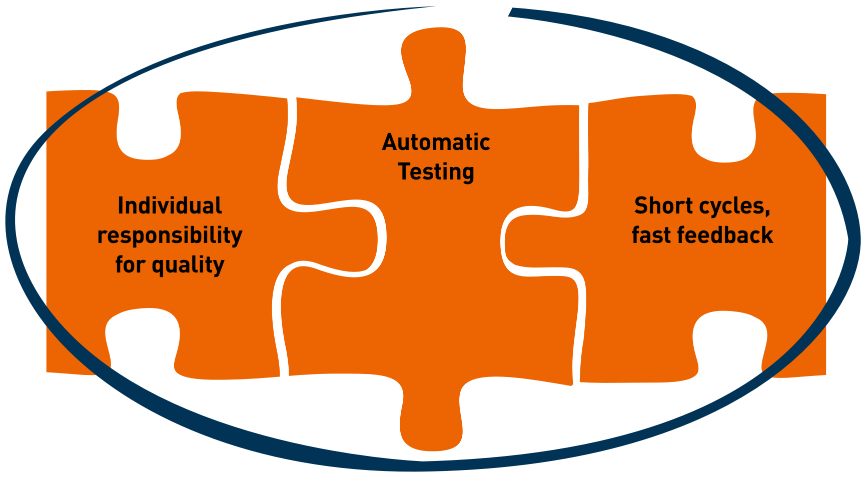
WHAT IS CONTINUOUS DELIVERY?

Continuous Delivery, CD, is a method that enables software organizations to deliver new versions or updates quickly and continuously without suffering internal delays or quality problems.

The method helps teams to collaborate on a common software product or service. It is based on a workflow where each developer takes great individual responsibility for the quality of the code. Automated testing is performed immediately after every change or addition to the code. The work is characterized by short cycles and rapid feedback. It means that the software is continuously tested and ready for delivery. New functionality, enhancements and bug fixes don't have to be packaged in major updates, they can be delivered as soon as they are ready.

The client can be a team in the same organization, or a business partner (who is not an end user of the software).

Constantly deliverable software


Automatic Testing
Individual responsibility for quality
Short cycles, fast feedback

Please note that the final deployment is not included in the concept of CD. When we make packaged and delivered software available to the end user, we would rather call it Continuous Deployment.

THE PRINCIPLES OF CONTINUOUS DELIVERY

Continuous Delivery means that the code produced is continuously under quality control at all levels, mainly through automated testing. Code should be tested in context and packaged to be ready for delivery as soon as possible.

Individual responsibility for quality
The main loop of CD is the individual developer's work with code changes that are built and tested before they are integrated with the team's other code changes. The work is based on trust and teamwork: each developer must carry out their part of the work in a predictable and accurate manner, without taking shortcuts or doing "kludge" (try reading Clean Code in five minutes).

Automatic quality assurance
To make quality assurance flow it must be automated as much as possible, even if the code review still has its place. Another advantage is that developers spend minimal time on the quality control of each delivery. Automation also reduces the risk of errors. Applying CD should speed everything up considerably. For example testing new code changes should be completed within one hour.

Short cycles with fast and continuous feedback
Here, CD follows the usual principles of Agile. CD is about getting complex collaboration to flow without delays or faults – like the cogs in a well-oiled machine.

By continuously adding small changes which are tested, teams get good insight into how far they have come, and can take a small risk with each addition. The product will most likely be able to go into production shortly after the latest amendments are made to the code.

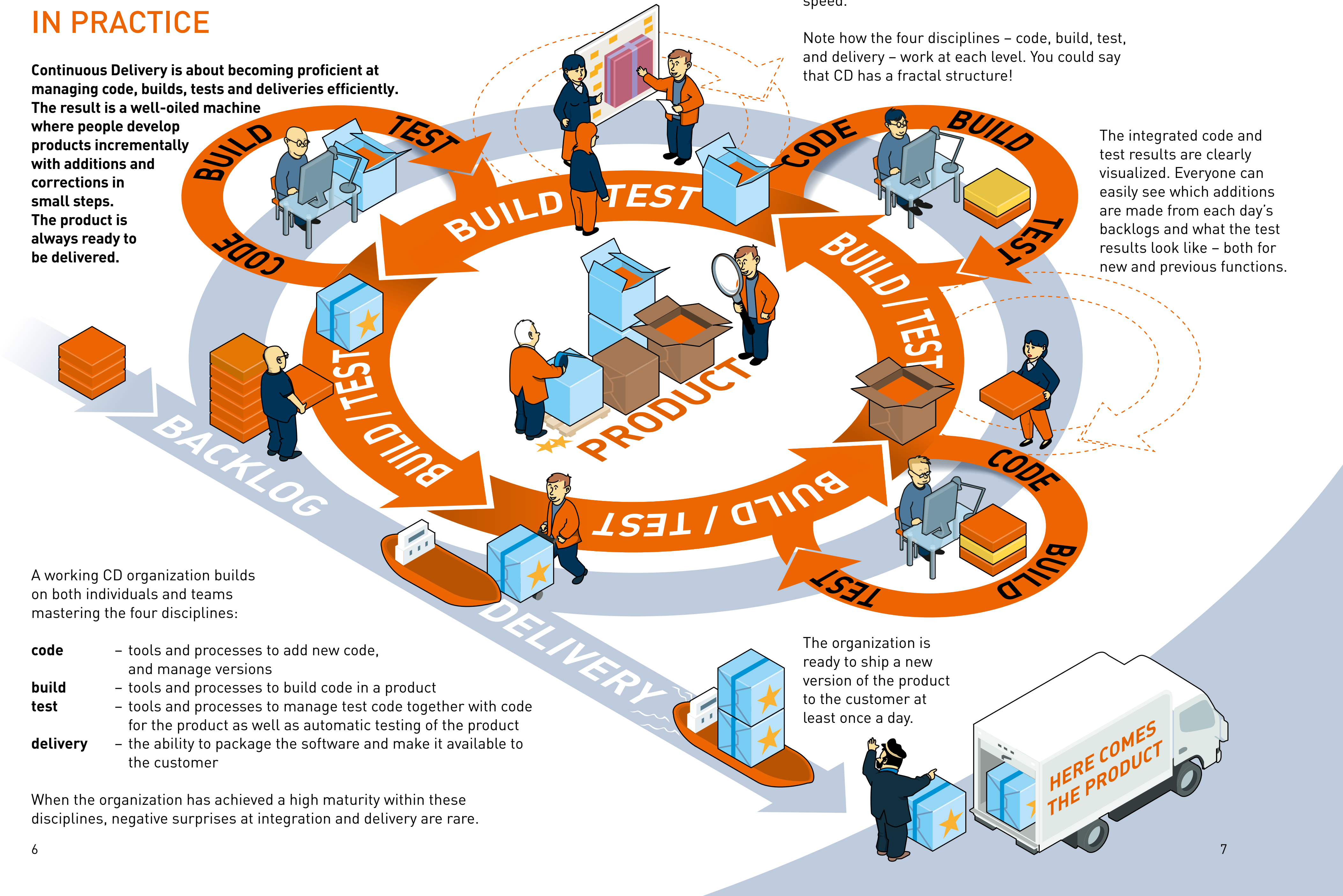
CONTINUOUS DELIVERY IN PRACTICE

Continuous Delivery is about becoming proficient at managing code, builds, tests and deliveries efficiently. The result is a well-oiled machine where people develop products incrementally with additions and corrections in small steps. The product is always ready to be delivered.

In a well functioning CD-organization the delivery wheel (= gray circle) spins at continuously high speed.

Note how the four disciplines – code, build, test, and delivery – work at each level. You could say that CD has a fractal structure!

The integrated code and test results are clearly visualized. Everyone can easily see which additions are made from each day's backlogs and what the test results look like – both for new and previous functions.



The organization is ready to ship a new version of the product to the customer at least once a day.

A working CD organization builds on both individuals and teams mastering the four disciplines:

code	– tools and processes to add new code, and manage versions
build	– tools and processes to build code in a product
test	– tools and processes to manage test code together with code for the product as well as automatic testing of the product
delivery	– the ability to package the software and make it available to the customer

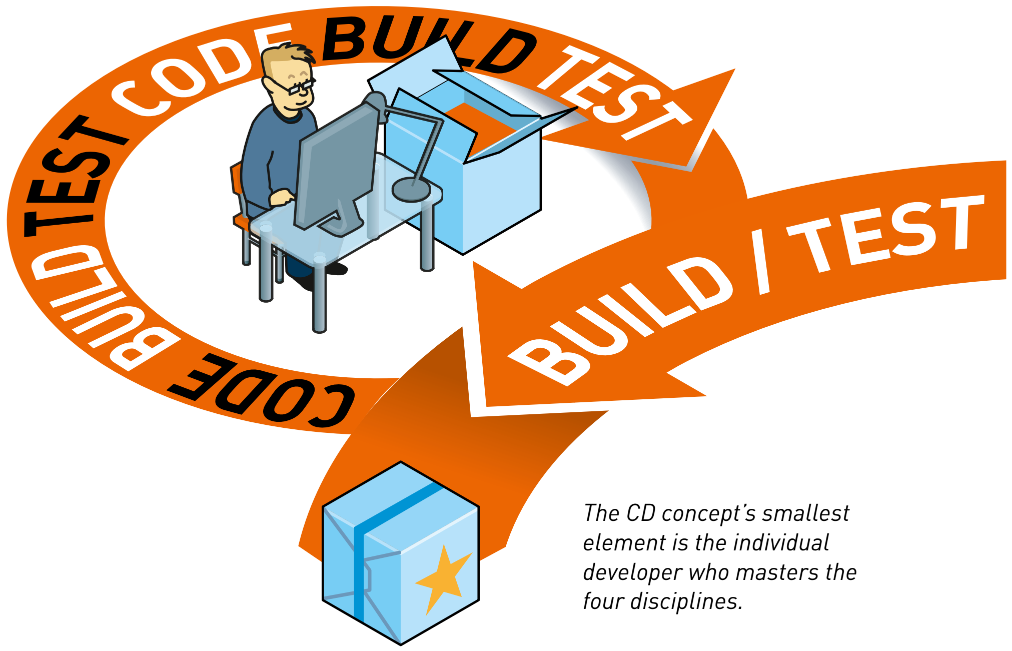
When the organization has achieved a high maturity within these disciplines, negative surprises at integration and delivery are rare.

PUTTING THE DEVELOPER AT THE CENTER OF CONTINUOUS DELIVERY

The aim of Continuous Delivery is to give the developer the most efficient possible tools and processes to enable management of the three disciplines code, build and test. The possibility to gain valuable feedback allows the developer to work in fast cycles and apply the fourth discipline: deliver code changes as soon as they are ready.

When everyone works in the same way, the latest code versions from other developers are available all the time – tested and ready. Waiting times between different parts of the organization become as short as they can possibly be. Through clear visualization everyone in the organization understands what is completed and how well different parts of the code work – both individually and when integrated.

A rule of thumb for CD says that a single code delivery should not take more than an hour. Build and automated tests will quickly provide feedback if delivery does not work as intended.



The CD concept's smallest element is the individual developer who masters the four disciplines.

The four disciplines

Code
Tools and processes to add new code and manage new and old versions of code that are developed and delivered. Adding new code triggers code reviews, builds and automated testing for direct feedback to the developer before the change is finally added to the product. What has been added to the product is made visible to everyone involved. Documentation of the addition is produced automatically.

Build
Tools and processes for building robust and efficient code in a product to give an identical result for everyone involved. Creates the opportunity to recreate old versions of the product.

Test
Tools and processes for managing test code together with product code product as well as automatic testing. Automatic testing continually visualizes how old and new product features function together. The developer has visibility about how each addition works before it is finally added.

Delivery
The ability to package software and make it available to the customer. In many cases it is appropriate to describe the configuration and setup of the production environment together with the product and tests.

GETTING STARTED WITH CONTINUOUS DELIVERY – 1

The method behind Continuous Delivery can't be implemented overnight. Instead it's a work in progress where the organization goes from novice to master. The central tool is a maturity model.

1
Analyze the business needs.

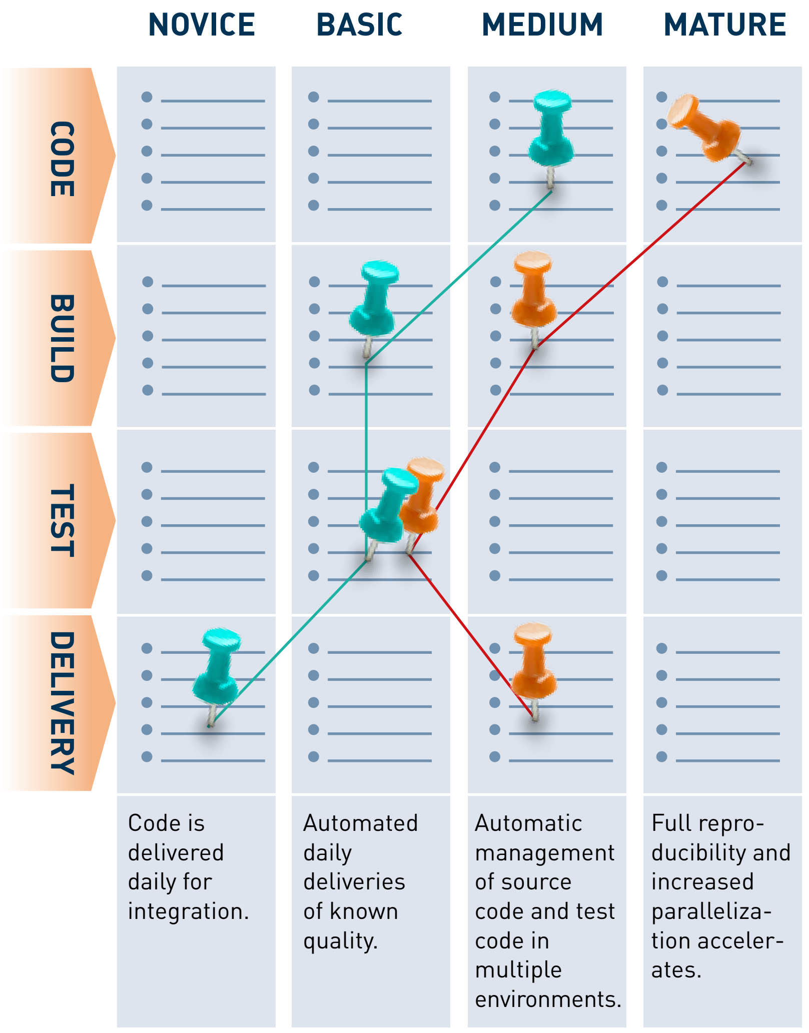
A well-functioning organization for CD is based on a deep understanding of the market and customer needs and business opportunities that the method creates. All decisions and priorities which are then made are based on this.

2
Introduce a maturity model and evaluate the organization's conditions and maturity.

A maturity model is a tool that both broadly and in detail shows how far the organization has come in terms of infrastructure and method. It forms the basis for assessment of the four different disciplines (code, build, test, delivery) before implementing CD. Even after you have taken the first steps, the maturity model is a key tool (see point 4 below); see it as a kind of back-log to improve and introduce new methods and processes.

3
Make a plan for improvements

Establish a plan based on the business analysis and insights from step 2. What actions are needed? In what order is it best to implement changes? Where can we quickly see the effects of any action? When can we start taking real advantage of the benefits of CD?


NOVICE
BASIC
MEDIUM
MATURE
CODE
BUILD
TEST
DELIVERY
Code is delivered daily for integration.
Automated daily deliveries of known quality.
Automatic management of source code and test code in multiple environments.
Full reproducibility and increased parallelization accelerates.

After analysis, evaluation and planning, just roll up your sleeves and begin to move through the maturity model from left to right. In the image we see the two organizations – the red and green – working through the maturity model. To go from "Novice" to "Mature" can take months, even years. Note that an organization can have different stages in the various disciplines.

GETTING STARTED WITH CONTINUOUS DELIVERY – 2

4
Implement improvements according to the plan

Implement improvements step by step according to plan and continuously evaluate if they are going in the right direction. The plan involves both investments in tools, training and processes within the various disciplines for managing code, build, test and delivery.

This work involves adapting and extending infrastructure. First and foremost, it is about bringing order to source code management, not least version management. The big challenge is to automate build and quality control. It is also important to visualize all processes and deliveries. Finally, you have to create a mature and automated delivery process.

5
Trim processes and introduce new ways of thinking

Once the infrastructure is starting to take shape it's time to align processes and move to the right in the maturity model. It also means introducing new ways of thinking:

Method thinking should take note of Principle 1 in the Agile Manifesto: "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software." It means that each completed task will provide a deliverable result that can be the basis for timely feedback in a simulated or real production environment.

Communication must be intensified at all levels, both within the development and between the customer and the development team. If the customer wishes, they can monitor how things are progressing day by day – total transparency is the goal. Developers may need to get more knowledge about how, where and by whom the software will be used.

Effective **leadership** in a CD environment is very much about strengthening trust within the development team – everyone should know that they are doing the right things the right way and that altruistic behavior is rewarded. Management must also develop a new boldness when it comes to exploiting the transparency, tempo increase and delivery reliability which CD creates.

Agile Manifesto:

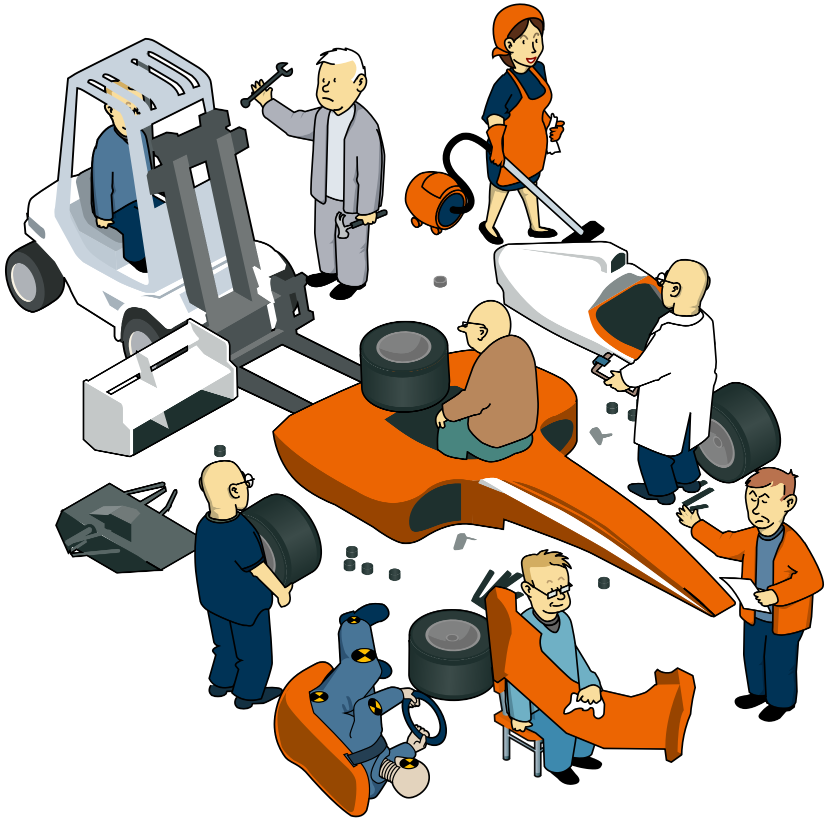
“WE FOLLOW THESE PRINCIPLES: OUR HIGHEST PRIORITY IS TO SATISFY THE CUSTOMER THROUGH EARLY AND CONTINUOUS DELIVERY OF VALUABLE SOFTWARE ...”

ANOTHER WAY TO EXPLAIN CONTINUOUS DELIVERY

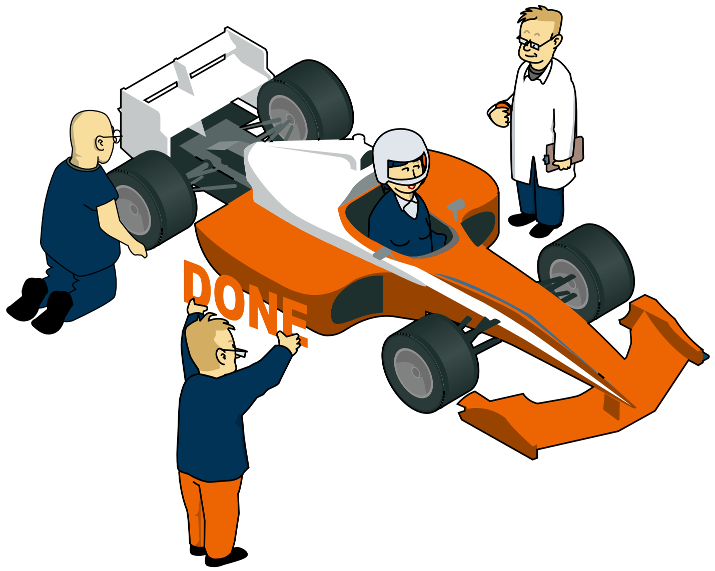
Batch Automobiles and Continuous Cars are two racing teams that compete in the Formula Agile Series. Competition is fierce, and each car's technical marvels must be modified and updated before every race. Batch Automobiles has long dominated, but now they have no chance against the newcomer Continuous Cars. How do these new stars get their cars perfectly adapted for each race?

Let's see what goes on behind the closed garage doors of the two racing teams.

[A] At Batch Automobiles the mechanics team gets to the bottom of the whole design at each update. They dismantle the car and work with each respective part. When everyone is done with their work, they build a functioning car again. Unfortunately, unexpected problems usually arise when they rebuild the car because the different parts have not been tested together. While the work is ongoing the drivers for Batch Automobiles know that they often have to wait several days until they get the chance to test drive the car and give feedback.



[B] At Continuous Cars they only remove the parts they need to make changes to. As soon as a mechanic makes a change they put back the part and test it so that it works with the rest of the car. Whenever the mechanic takes a break the drivers get in the car, drive a few laps and provide feedback.



Continuous Delivery – puts you closer to your customer

Continuous Delivery, CD, is a method based on:

- **Individual responsibility for quality**
- **Automatic testing**
- **Short cycles of rapid and continuous feedback**

This creates many advantages for the development team in the form of higher productivity, less waiting, fewer compatibility issues between new and old software, and simplified troubleshooting. But the big winner is the customer who receives faster delivery of code that maintains a higher quality, and is better adapted to the market's rapidly changing requirements.

Softhouse Consulting

Stockholm
Tegnérgatan 37
SE-111 61 Stockholm
Phone: +46 8 410 929 50
info.stockholm@softhouse.se

Göteborg
Kungsgatan 19
SE-411 19 Göteborg
Phone: +46 31 760 99 00
info.goteborg@softhouse.se

Malmö
Stormgatan 14
SE-211 20 Malmö
Phone: +46 40 664 39 00
info.malmo@softhouse.se

Karlskrona
Campus Gräsvik 3A
SE-371 75 Karlskrona
Phone: +46 455 61 87 00
info.karlskrona@softhouse.se

Växjö
Willans Park 3
SE-352 30 Växjö
Phone: +46 455 61 87 00
info.vaxjo@softhouse.se

Sarajevo
Džemala Bijedica 185
BA-71 000 Sarajevo
Phone: +387 64 42 79 847
info.bosnia@softhouse.se

www.softhouse.se

SOFTHOUSE