ADNOVUM

DIGITALIZATION OF SOFTWARE ENGINEERING

Everything uses software nowadays. Availability, stability and flexibility demands are constantly increasing, making more efficient production methods necessary. The software industry is now about to take a step most other industries have long behind them – it needs to digitalize processes.

By Remo Meier and Stefan Ott

Faster, further, higher. As with athletes, companies are being conditioned for high performance. Efficiency and productivity need to increase, costs need to fall, product changes and new products need to be implemented and launched as quickly as possible. Therefore, procedures and processes need to be optimized at all levels, by harmonizing the transfer of semi-finished products across the production phases.

One solution is obvious: reaching these goals with IT. Many industries are fundamentally changing their relationship to IT. IT procurements are no longer fixed procurements like those for water and electricity. On the contrary, IT is becoming an integral part of a company. As such, IT is moving ahead faster and further in tandem with the companies.

EXTERNAL INNOVATION CAN CREATE EXPONENTIAL GROWTH AND SUCCESS.

The new relationship to IT means industries are focusing increasingly on successful IT concepts such as mobile apps and the Internet of Things (IoT). The concepts are opening new opportunities. Companies can expand products into platforms by making the interface available to end customers and third-party companies. Doing so allows for external innovation, which can create exponential growth and success.

In short, more and more software is needed. The demand creates several challenges for IT companies: They need to link the development of software from design to operations, improve and accelerate production, eliminate media disruptions and allow for rapid exchange of information in order to meet market demands.

From project to solution

The increased deployment and higher visibility of IT results in the desire for a new cooperation between the user/customer and their IT service providers. The focus is on closer and more efficient contact, which extends from analysis and conception to production and further development. Today, the typical customer is different. A few years ago, the internal IT department was usually the customer of the service provider. But today, another company department may order an application. The customer has numerous ideas and wants to integrate them. In order to evaluate and comment on them, faster solutions are needed. The same applies to changes. They need to be taken into account as quickly as possible, so the IT provider can meet customer needs. In such a context, the limited project is transformed into a solution with a life equivalent to the entire software life cycle and thus typically years. Software as service – in the best sense of the word. The consequence: The demands on the inte-gration and quality of the software increase.

THE FACT THAT THE SOFTWARE INDUSTRY CAN FACE THE CHALLENGES USING ITS OWN RESOURCES IS A DECISIVE ADVANTAGE.

From digital production island to digital assembly line

What is the next step now? The focus is on process automation. The fact that the software industry can face the challenges using its own resources is a decisive advantage with enormous potential. Methodologies and expertise are available. The industry is now required to apply the knowledge to their own processes to redesign and automate them.

Stefan Ott (left) and Remo Meier describe how automation speeds up development of custom software.

The significant factor here is that the life cycle of software involves a lot more than development. The most important factor is to create uniformity and consistency and to do so with design, development, testing and production and back again. The following measures can help reduce the throughput times: automating deployment, establishing and coordinating technological and organizational measures for zero downtime, employing risk management, ensuring service availability and performance, tracking performance and last but not least guaranteeing quality. Integration of operations and preventative measures should help reduce required maintenance to an absolute minimum. Collaboration platforms, for example, remove barriers between the relevant organizations, ease the exchange of information and make information available at any time.

The expansion of code

In doing so, we do not have to start from zero. Software engineering services have been undergoing automation for years. A quick look at the code supports this. The code is more than just the actual application code and the code for the system. Documentation is stored as code with markup languages. Large parts of the infrastructure in servers, networks and storage are defined using code and also automatically created and maintained (Infrastructure as Code, DevOps). Parts of the integration and testing are also automated and transformed to code. And there is more: the fully automated software deployment – without downtime and under integration and fully automated control of load balancers – or the elastic scaling of systems such as servers, applications and nodes based on the load behavior visible in the monitoring system (automated feedback).

The code is more than just the actual application code and the code for the system.

Code to create reproducibility

Having the entire IT system in the code opens up numerous new possibilities. Code has thus been installed in version management systems for years. Doing so makes archiving possible along with access to and comparisons among versions as well as the reset to previous states. An entire system copy can be created, developed and then be integrated back into the primary development branch. The development branch thus remains fully functional and additional changes can be integrated quickly into the production at any time.

The elimination of all other influences is good for the consistency and the automation of development. Processes and the systems status should possibly be the same in all environments. This eases detection of problems early or even helps avoid them entirely. Approaches such as immutable infrastructures integrate this concept: Servers are thus no longer installed in the classic sense, but final images are created from code, and these cannot be changed. The images are automatically delivered to, installed on and run in the server. The process is identical for test and production environments. The configurations receive the images from their surroundings. The data entry from business records to auditing and logging is performed in central data bases. The static nature of the images allows for multiple infrastructure simplifications. Aspects such as firewalls, remote access, user controls and file system authorizations are thus irrelevant in many areas. Precisely the same system can be recreated and started once more from the code, just like software can always be created anew again and again. Every step in the system is reproducible. No manual steps are needed. Doing so increases efficiency, quality and availability. A system is always precisely defined by its code. And an individual version number is sufficient to identify the code.

Every step in the system is reproducible. No manual steps are needed.

From servers to events

Why stop here? Maintaining the server landscape is associated with significant expense. Once the system is defined in the code and further influences have been eliminated, automation will be able to make a significant step forward. To what extent is a server you have installed, started and maintained still necessary, if the server can be generated at any time from the code, in the most extreme form even for every new query or batch process? Instead of servers, we refer to events. The advantages of the event approach are obvious:

- No installation and starting of servers required.
- Infrastructure will only be claimed when used and can be scaled for the application.
- Redundancy and error tolerance are available automatically and transparently. Events are distributed dynamically to the machines based on the current load, geographic distribution, available hardware and the code performance requirements.
- Server housekeeping is no longer needed.
- No more server processes requiring maintenance over months and years are needed.

Various factors influence the successful application of the event approach. For example, consistent and, in many areas, automated monitoring is vital in daily operations and easy to implement. The scaling from production to development and testing are essential both for efficient development and for local traceability of the system by the developer. In conjunction with permanence and reproducibility, these factors not only allow the monitoring and overview but also avoidance of many problems – relative to traditional solutions.

Successful code is simple code: text files, which both machine and people can read and edit.

Consistency

A successful code is a simple code: text files, which both machines and people can read and edit. Code can be used for more than implementation and deployment; it can also be included in analysis and design. Doing so creates consistency across the entire product life cycle.

Thus larger IT projects now need to have a business analyst on hand. They know what is already in place, ascertain the (new) requirements, design and document a solution. The duties will generally remain the same. The procedure and the type of output will change massively, as the "classic" approaches are associated with media disruptions. The media disruptions prevent speedy and proper information management, which is vital in order to meet today's demands for efficiency and automation.

The decisive step is now to save the concept and the information model for the business analysis (with use cases, user interfaces, etc.) as a data model. Such a model already exists – at least in the minds of business analysts. The advantages are obvious:

- Information entry is structured: Each piece of information belongs in a specific location and is not maintained redundantly.
- The recorded information is up to date and immediately available in the model: Those who want access to certain data, whether they are the customer, the software team or the operation team, can have it in the desired form at any time. In addition, information also flows back as comments and feedback.
- Relationships/links between concepts may be generated and tracked, which contributes to a coherent overall image of the output.
- Further developments can access the available information. In addition, changes to the information can also be recorded and then links used to avoid gaps and unintended consequences.

The information model forms the basis for project planning and methodologies. Their concepts can be defined as tasks. Links can be used to gather all the necessary information for a task. Doing so allows for a better expense estimate and thus better tracking of the project. Cockpits help in the process as they provide an overview of the progress of the development. The information is then projectbased. The expert information acts purely as a reference to the information model.

Alternatively, domain-specific languages (DSL) can be used in order to model the system in code. The model describes components, their interfaces and interactions. The model can be stored in repositories and made accessible online. Models in this case are not limited to the description but also serve as a starting point for the creation of software components. They can be updated or transferred to newer models in the later phases of a project. In the event of problems they provide valuable feedback regarding the relevant use cases.

The findings

The next steps and results of software engineering always remain in the code, however, with the focus on digitalization and automation. This creates new challenges, particularly for architecture and design. All of these tasks require software engineers who are willing to strike in a new direction and able to work in a team. ■

Remo Meier

Remo Meier, MSc. and Dr. sc. from ETH Zurich, joined AdNovum's Application Engineering team in 2013. The department focuses on providing the means for efficient software development. In this context, topics relevant to the digital evolution are very important. When Remo Meier has free time, he can be found in the mountains.

Stefan Ott

Stefan Ott, MA in Information System Management and HSPTP, has been a Senior Business Analyst with AdNovum since 2007. He sees innovation not only in contact with customers. He is also part of an in-house team to expand services in the business analysis team. His duties include training new employees, defining business analysis processes and the further development of deployed tools. One of his hobbies is running marathons, where he has plenty of time to develop ideas and think of optimizations.