IoT-DRIVEN BUSINESS INNOVATION

LESSONS LEARNED – SOFTWARE DEVELOPMENT IN THE ERA OF IOT

When comparing IoT (Internet of Things) software development with traditional IT environments, it becomes clear that developers in the era of IoT face new technical and methodological challenges which are nonetheless hard to detect at first glance.

Concept Reply has already demonstrated its expertise in numerous IoT projects. Read about the seven lessons the IoT specialists of the Reply Group were able to draw from their experiences.

IOT SOFTWARE DEVELOPMENT IS THE SOFTWARE DEVELOPMENT STRATEGY OF THE FUTURE.

If IoT projects are approached properly with a competent development partner at the customer's side, the quality of the software increases, iterations are avoided, (subsequent) development costs are lowered, customer friendliness is increased, and maintenance costs are reduced.

The Internet of Things has long been a reality. 27 billion devices were connected via the IoT in 2017. This number will likely swell to 125 billion by 2030. By 2022, business related to IoT products is expected to comprise a market worth 561 billion US dollars. This trend towards increasingly networked products entails a few changes for software development. When comparing IoT software development with traditional development scenarios, it becomes clear that developers in the era of IoT face new technical and methodological challenges which are hard to detect at first glance. First of all, ensuring the quality of the software can quickly become a challenge. Incorrect architecture decisions lead to security risks and software that is hard to maintain. This results in high costs and reduced competitiveness.

But despite all the obstacles and challenges, IoT projects are the solution for viable software development and will soon be unavoidable. Concept Reply has already demonstrated its expertise in numerous IoT projects. In the process, it has gained a series of important insights that can help overcome these challenges.

27 billion

devices are connected via the Internet of Things in 2017.

125 billion

devices are connected via IoT by 2030.

In 2022

IoT generates business worth $561bn.

LESSON ONE: SECURITY MUST BE INCORPORATED INTO THE ARCHITECTURE FROM THE BEGINNING.

TRADITIONAL: Whereas in traditional software development it is often possible to make an application secure later on by releasing a new version, in an IoT environment it sometimes happens that devices can no longer be updated with improved software at a later date. In that case, the user's only options are to live with the security hole or replace the device with a new one.

IoT: An IoT environment is extremely complex, because many components intercommunicate and exchange data within it. In the short term, security holes can often only be resolved on the gateway side or by rectifying the weak point directly. At the same time, there are special requirements on the device side (for example, special micro-controllers or limited memory), which means fixes are not readily available. An update option for devices must be planned from day one using over-the-air update functionality. But in the event of a breach, it is also helpful to have a software architecture that is already designed to block or limit access to gateways or the app server.

LESSON TWO: DEVICE SIMULATORS HELP IN EARLY PHASES AND DURING TEST AUTOMATION

TRADITIONAL: Whether apps are designed for the Web, smartphones or desktop computers, they all have a client-server architecture. The client runs on the front end and the server on the back end. Whereas the client maps a local state and displays a section of the global state, the server is in a global state that can be changed by the clients. Therefore, testing processes can be limited to these small quantities of states and interactions.

IoT: In addition to the front end on the client and the back end on the server, the edge software is implemented on the IoT gateway, which is responsible for the Internet connection and simple data processing as well as the 'thing software' on the actual device, such as a sensor. This increases the complexity of the system, which is determined based on the number of states (which are dependent on the physical environment) and possible interactions, among other things. The increased complexity necessitates more testing work. Because the actual devices are usually not yet available in early stages of a project, device simulators are an indispensable tool. They help with automated testing in the cloud, among other things, but they also simplify load testing. Devices that are already available can of course be integrated into the test automation, but this makes testing in the cloud more difficult; after all, the devices are usually located behind a firewall in the company network. For this reason, it is necessary to create a complex test infrastructure that differs from traditional test infrastructures.

AN EXAMPLE: In a customer project, Concept Reply was tasked with using end-to-end tests to test the correctness of the implementation of IoT software functions, some of which run in the cloud (e.g. long-term data storage and generation of alarms), some on field devices (e.g. recording of sensor data) and some on end devices such as smartphones and laptops (e.g. user management and changes to settings). The test measures one or more test cases to determine which IoT software functions should be performed as part of the testing. This measurement is difficult, however, because the functions are distributed among different platforms. This is easier in traditional software projects, because the platforms are homogeneous.

LESSON THREE: OTA UPDATE FUNCTIONALITY IS A SIGNIFICANT COMPETITIVE ADVANTAGE.

TRADITIONAL: In traditional IT, users install new software, usually by themselves, or they receive information that a new version is available. The user therefore decides on their own whether or not to install an update. The disadvantage: Updates that are important and, in some cases, critical for security may easily be overlooked.

IoT: As the number of devices and equipment in an IoT scenario increases, it is hardly possible any longer to maintain an overview of the situation and install such updates manually. The deployed devices can often only be kept up to date via over-the-air updates (OTA updates). This has to be factored in from the beginning – also with regard to the overall application lifecycle. Devices that have security holes, but no update mechanism must be discarded. Another advantage: An update mechanism enables new business models. Whereas at the start of a project a customer often has no idea what functions his development process will need to implement in future, an integrated update mechanism allows software functions that do not yet exist to be distributed at a later date.

LESSON FOUR: TROUBLESHOOTING REQUIRES COMPREHENSIVE LOGGING FROM THE DEVICE ALL THE WAY TO CLOUD!

TRADITIONAL: In a traditional IT setup, there is generally an app server, possibly several clients and logs are readily available on both sides, so that errors are easy to diagnose.

IoT: In the world of IoT, the situation is somewhat different. There is a large number of devices and possibly intermediary gateways. Searching for the cause of an error is quite a bit more complex here than in a traditional software environment. Logs therefore have to be made available centrally, partly because under certain circumstances there may be space limitations on the device side. A central log database (or rather syslog-ng) is a possible solution to the problem of high device numbers. Battery operation is another problem. This can be solved by having the device send status information to the cloud every time it goes online.

LESSON FIVE: AGILE METHODS OFFER THE REQUISITE FLEXIBILITY FOR IOT INNOVATIONS.

TRADITIONAL: Traditional software development based on the V or waterfall model is slow and inflexible. It calls for requirements and features to be determined prior to development. Once the specifications have reached maturity, the next step is to implement precisely those specifications. This model makes no provisions for changes or extensions in the implementation phase, which means that innovations cannot be incorporated in the development process at short notice. There is thus little room for quick modifications. For longer-term development projects, the software is often already outdated upon completion, because processes or requirements have changed in the meantime.

IoT: The much-discussed 'customer experience' is not only a decisive criterion for success in the front end, i.e. in apps and websites; it also plays a role in devices in the form of user-friendliness. For many customers, user-friendliness often plays no role at first, which is a mistake, because it is important for the long-term success of the solutions. During the development stage, the developer should, for example, already consider how the setup process for an IoT gateway should look. Because IoT projects are often innovation projects, one of the customer's top priorities is often to achieve a business impact with them, for example, by reducing maintenance costs or marketing software functions in general. Because the hoped-for business impact is difficult to assess, especially in the early stages, it is important to rely on an agile approach. This allows the business case to be developed in parallel. The only remaining challenge then is to stay focused and not get lost in unnecessary features.

LESSON SIX: CLOUD NATIVE IMPROVES TIME-TO-MARKET.

TRADITIONAL: In a traditional architecture, there is no IoT hub, which usually consists of a message broker, message buffer (if devices are not online), device shadow (for storing the device status), device provisioning (for connecting new IoT devices to the back end) and device management (for managing existing devices, e.g. firmware updates).

IoT: Cloud native IoT hubs come with the required functionality already integrated out-of-the-box as standard and can therefore be set up with little effort. They are also relatively easy to upscale if a large number of devices is needed. SDKs, which are offered for many software versions, also help accelerate software development. The alternative to cloud native IoT hubs are IoT hubs developed in-house (based on Kubernetes, open source products and individual software). Proprietary IoT hubs can be designed in such a way that they can run in different cloud environments (e.g. AWS or Azure). This makes the customer independent of cloud providers, but they do then face many questions as a result: Should we handle everything on-premise, off-premise or even cloud native? Should open-source tools be used or are proprietary software solutions preferable.

THE FOLLOWING HAS TO BE CONSIDERED:

- **On-premise** offers low latency combined with extremely high data rates (but be careful, because this can lead to higher connection costs). An on-premise solution can be implemented in a data centre as an individual IoT gateway or as a cluster, and there are existing frameworks for the individual IoT gateways (e.g. AWS Greengrass, Azure IoT Edge SDK, Eclipse Kura, Node RED and much more). However, a standard has yet to emerge, and hardware providers from the industrial sector predominate here. Their solutions also take into account the requirements relevant for their sector, such as ruggedness (e.g. shock or temperature resistance). Furthermore, security has to be guaranteed in the field (TPM stores are a good approach for storing certificates in the field).

- **Off-premise** customers can choose from cloud native services such as AWS Lambda, AWS DynamoDB or Azure IoT Hub.

- **Cloud native solutions** simplify the work of the developers and users and shorten the time-to-market. But cloud native also increases dependency on a cloud provider. Hybrid strategies are an interesting compromise (parts in cloud native, other parts in docker, etc.). Cloud native solutions can, in principle, be

replaced by open source products (e.g. Kafka instead of AWS Kinesis). What's more, cloud native offers a comprehensive security concept. For example, AWS Lambda functions can only access authorised SQL tables; it usually takes time and effort to develop a comparable security concept with open source solutions.

AN EXAMPLE: For one of its customers in the field of industrial automation, Concept Reply relies on a Kubernetes cluster to perform and provision back-end functions. The cluster requires reserved computing and storage capacity, regardless of whether the back-end functions are used. This reserved capacity automatically generates operating costs that are not incurred by cloud native solutions. Cloud native is therefore interesting at early stages when the products are not yet heavily used. It is also a good choice when usage fluctuates greatly over time, that is, when steady utilisation throughout the day is not expected.

LESSON SEVEN: EMBEDDED COMPONENTS SHOULD NOT BE DEVELOPED IN ISOLATION.

TRADITIONAL: The traditional approach to developing embedded components is to develop the hardware for a predefined function. It therefore makes sense to follow the principle: "As specified as possible, as little as possible". But this approach runs into problems when a platform is to be expanded or updated with features in future, because new features may place heavier demands on the hardware.

IoT: Conflicts arise here among issues such as energy efficiency, availability and performance. An example: A device that has to cover an extensive range also needs a lot of transmission power, which can only be maintained temporarily due to high power consumption. However, the dimensions of a hardware component are determined based on specific requirements that offer little room for the requisite flexibility if all of the aforementioned criteria are to be met (energy efficiency, availability and performance). It is therefore important to plan in advance. The most important questions here are: Can requirements be implemented in the future as well with the planned hardware, and how much flexibility is required for this?

CONCEPT REPLY

Concept Reply is the software development partner of the Reply Group specialised in IoT innovation and offers solutions for its customers in the areas of Smart Infrastructure, Industrial IoT and Connected Car from the idea through the concept phase to implementation, operation and support. With around 50 IoT specialists, the range extends from implementation in the embedded environment to gateway software or cloud applications.