Towards understanding the impact of requirement evolution on deployment

Florian Schmalriede and Andreas Winter
Carl von Ossietzky Universität, Oldenburg, Germany
{schmalriede,winter}@se.uol.de

Abstract

Distributed and heterogeneous systems, such as IoT systems, enable many different alternative software deployments which lead to different system characteristics. Evolving functional or non-functional requirements might lead to adapting the deployment. This paper shows a case study where changed requirements result in a new deployment, motivating early deployment planning in software evolution.

1 IoT Systems Software Deployments

Internet of Things (IoT) systems integrate real-world things into digital networks, allowing computer systems and users to interact with them without direct physical contact. Computer systems and users analyze properties of things and influence them according to use-case-specific business logic. Processes become fully or partially automated.

IoT systems are realized as distributed and heterogeneous systems in which subsystems with different profiles cooperate across different network technologies to fulfill application-specific goals. Sensor and actuator nodes, directly attached to things or in their environment, digitize and influence properties of things via physical interactions. Servers process gained information and respond accordingly to outcomes automatically. Clients present gained information to users and react to input from users accordingly. Gateways intermediate between environment-dependent and device-specific network technologies and link individual networks of sensor and actuator nodes, servers, and clients into collaborative networks.

Different subsystems in IoT systems vary e.g. in terms of their computing power, storage capacity, power consumption and communication bandwidth due to their environment or profile. Neverless, often all subsystems are suitable to deploy software that provides requested functionality. Accordingly, there is a wide range of alternative deployments in IoT systems.

Depending on software deployment, subsystems of IoT systems have different workloads. This implies that different software deployments affect the characteristics of IoT systems in different ways. For example deployments which will lead to more workload on mobile subsystems is likely to reduce their runtime. Accordingly, deployment of software has impact on the fulfillment of requirements that specify which characteristics need to be met. Vice versa, requirements restrict the range of possible deployments. Changing requirements over time may lead to changing software deployments. If this hypothesis is confirmed, it becomes relevant to consider impacts of requirements evolution on software deployment to address them in a deployment planning phase.

To determine, if evolving requirements will impact the deployment of software, three case-study experiments are performed. The first experiment shows that initial requirements can be fulfilled with an initial deployment. In the second experiment requirements from first experiment evolve and it is shown that the initial deployment from first experiment do not met the new requirements. The third experiment show that evolved requirements can be met with a different deployment.

2 Case Study

Monitoring cold chains (cf. [1, p. 28ff.]) for food during transport is selected as an application scenario for IoT systems. Foods are placed in transport boxes, each equipped with a temperature sensor node, and loaded into cold trucks. For independence, temperature sensor nodes are supplied with mobile power sources. There are fixed gateways in cold trucks, powered by cold trucks battery, to which temperature sensor nodes share food temperature. Gateways establish an internet connection and forward measured temperature values. Additionaly, gateways can interact with drivers in order to notify them in cooling issues. Thus, gateways serve as gateways and clients at the same time. A server receives information about cold chain via internet and stores them for later access. With clients, contractors and customers access information on the cold chain on the server.

Since performing experiments with real cold trucks is very complex, a model[1] was used in [3] to perform the experiments. Each of two temperature sensor nodes are realized by an ESP32 devkit-C32D developer board and a BlueDot TMP117 temperature sensor. Rechargeable batteries with a storage capacity of 65 $\frac{\mathrm{W}}{\mathrm{h}}$ are used as power source for the temperature sensor nodes. A Raspberry PI 4B with an attached 4 inch display is used as gateway. Temperature sensor nodes connected via WiFi, to a gateway providing a local hotspot. Only gateway and temperature sensor nodes are considered in the experiments, so server and clients are simulated together on a conventional laptop.

[1] The authors thank Phillip Bühring for building the model and performing experiments in his master's thesis [3].

Each experiment was performed over a period of one hour. Every second, a temperature reading was collected from both temperature sensors. Functional requirements were validated with acceptance tests, where collected temperature values were replaced with simulated temperature values to be comparable across experiments. Non-functional requirements related to the runtime of temperature sensor nodes were calculated using the capacity of mobile energy sources and the energy consumption of temperature sensors nodes (see table 1). Energy consumption was measured by INA226 power sensors which average 256 readings and have an A/D conversion time of 142 µs.

2.1 Experiment 1

Rudimentary cold chain monitoring requires functionality to detect and share cold chain interruptions. It is required that cold chain monitoring functionality is continuously available at least one working week, here Monday 6:00 am to Saturday 6:00 pm. Thus rechargeable batteries of temperature sensor nodes must last 132 hours. Between Saturdays 6:00 pm and Mondays 6:00 am, batteries can be recharged.

A cold chain monitoring software, that compare measured temperature of food continously against hard thresholds, detect cold chain interruptions on sensor nodes. If cold chain interruptions appear, the software shares information about them with other subsystems. Software on gateways inform drivers about cold chain interruptions and forwards information about interrupts to the server. The server stores information about interruptions and gives customers and contructors the possibility to consult them via clients. This system detects cold chain interruptions and prevents spoiled goods from entering the market.

The deployment in `Experiment 1` meets the functional and non-functional requirements. Acceptance tests were successfully passed and batteries run time of 141 hours exceeds the required 132 hours.

2.2 Experiment 2

The cold chain monitoring can be improved by predicting possible cold chain interruptions. If a cold chain interruption is to be expected drivers are informed such they can react. Runtime and charging time of rechargeable batteries remain unchanged to `Experiment 1`.

In order to meet emerging functional requirements, the cold chain monitoring software deployed on the temperature sensor nodes is extended by complex calculations to predict cold chain interruptions. Software on gateways are extended to inform drivers about upcoming interruptions. Beside detecting and reporting cold chain interruptions, the upgraded system detects and informs about upcoming cold chain interruptions.

The deployment in `Experiment 2` meets the functional but not the non-functional requirements. Acceptance tests were successfully passed, but batteries runtime of 105 hours is below the required 132 hours.

2.3 Experiment 3

Reworking the deployment in experiment 3 improves these disadvantages. The cold chain monitoring software is now deployed on gateways. Thus temperature sensor nodes do not have to perform complex calculations and save energy for measuring and sending temperature values, only.

The deployment in `Experiment 3` meets the functional and non-functional requirements. Acceptance tests were successfully passed and batteries run time of 149 hours exceeds the required 132 hours.

Experiment	Power consumption
`Experiment 1`	$\approx 460\ \frac{\mathrm{mW}}{\mathrm{h}}$
`Experiment 2`	$\approx 618\ \frac{\mathrm{mW}}{\mathrm{h}}$
`Experiment 3`	$\approx 437\ \frac{\mathrm{mW}}{\mathrm{h}}$

Table 1: Average power consumption of temperature sensor nodes, taken from [3].

3 Conclusion

By having an initial deployment that fulfill initial requirements and evolving the initial requirements that will not be fulfilled by initial deployment it is shown, that requirement evolution can have impact on deployment. Using an improved deployment which met the evolved requirements, shows that changed requirements call for revised deployments.

In order to avoid subsequent adjustments of deployments, including hardware adaptations, requirement evolution has to be considered already in the initial engineering process. Accordingly, deployment does not change during evolution of considered requirements.

More generally, the results show that requirements affect the deployment of software in IoT systems. Whereby different deployments may meet the same functional requirements, but may not meet same non-functional requirements. This motivates a more detailed research on the impact of requirements on deployments as well as deployment planning in order to develop requirements-aware IoT systems.

Beyond the results of the experiments, a realistic model for IoT systems was built in [3], which will be used for further research on the impact of requirements on IoT systems deployment. This allows to close gaps in research identified by [2] and will be used as a basis to find a structured way to plan the deployment of functionalities in IoT systems.

References

[1] A. Bassi et al. *Enabling Things to Talk.* Springer Berlin Heidelberg, 2013.

[2] A. Brogi et al. *How to place your apps in the fog: State of the art and open challenges.* Software: Practices and Experience, May 2020.

[3] P. Bühring. *Case studies in IoT-Deployment.* Master's thesis. Carl von Ossietzky Universität Oldenburg, February 2023.