In practice

A case study on the development of the German Corona-Warn-App☆

Mohamad Fawaz Enaya [a], Thomas Klingbeil [a], Jacob Krüger [b,*], David Broneske [c], Frank Feinbube [a], Gunter Saake [d]

[a] *SAP SE Walldorf, Germany*
[b] *Eindhoven University of Technology, The Netherlands*
[c] *DZHW Hannover, Germany*
[d] *Otto-von-Guericke University Magdeburg, Germany*

ARTICLE INFO

ABSTRACT

The COVID-19 pandemic has drastically changed daily life and required fast responses to new situations, such as restricted public life. A major means to limit infections have been contact-tracing apps that inform an individual about a potential infection, helping to initiate countermeasures faster. While different tracing apps have been compared technologically, we are not aware of studies providing insights into their development processes during the pandemic emergency situation. To address this gap, we report an exploratory case study on how the German open-source Corona-Warn-App has been developed at SAP SE—and how other organizations (e.g., Deutsche Telekom AG), researchers, and individual developers contributed. We elicited data on the process, practices, and challenges by interviewing six developers at SAP SE, analyzing documentation, and discussing our data with an expert on the app's development. Overall, we provide insights into how the development process of the Corona-Warn-App differed from other projects at SAP SE (e.g., testing), discuss the causes (i.e., public interest causing researchers to perform tests), and study the consequences (i.e., emergency tickets by researchers). Our findings can guide organizations when developing software in similar emergency situations (e.g., pandemics) in which reliable software needs to be developed within a short period of time.

1. Introduction

The global COVID-19 pandemic has caused a major health crisis that threatens the well-being, and thus impacts the daily life, of individuals all around the world. To cope with the pandemic, politicians enforced countermeasures (e.g., restrictions on public life) and researchers developed medical solutions (e.g., antibody tests, vaccines). A major means to tackle the pandemic contributed by the software-engineering community are tools that make individuals aware of their potentially infectious contacts to limit the spread of COVID-19. Particularly, *contact-tracing apps* have been developed, such as the NHS COVID-19 app in England, the Covid Alert app in Canada, the Aarogya Setuas app in India, or the Corona-Warn-App in Germany (Erikson, 2021; Munzert et al., 2021; Seto et al., 2021; Wymant et al., 2021). Such apps track infections and inform their users about potentially infectious contacts, helping to decide on proper countermeasures, such as testing or self-isolation (Abuhammad et al., 2020; Gupta et al., 2021).

Despite the immense public interest in such apps and their open-source nature in many countries, we are not aware of detailed reports on their development processes. Recent studies mainly focus on technical comparisons (Reelfs et al., 2020; Ahmed et al., 2020; Gupta et al., 2021), ethical and privacy concerns (Morley et al., 2020; Abuhammad et al., 2020), the apps' impact in practice (Garousi and Cutting, 2021; White and van Basshuysen, 2021; Wymant et al., 2021), or the fulfillment of user requirements (Sutcliffe et al., 2021; Bano et al., 2020). However, a more profound understanding of the development processes is important for practice and research alike. Regarding practice, experiences on best practices help prepare software organizations for future emergency situations (e.g., natural disasters, other pandemics). Regarding research, the emergency situation exhibits unique properties (e.g., apps for managing the pandemic, developing in a pandemic situation) that require further investigations, for example, to provide better automation or closer integration of different external stakeholders.

In this article, we report the results of an exploratory case study (Runeson and Höst, 2009) on the development of the German Corona-Warn-App, which was commissioned by the German government and developed primarily by SAP SE (software development) and Deutsche Telekom AG (infrastructure). We conducted six semi-structured interviews at SAP SE and inspected the Corona-Warn-App's documentation as well as version-control system to elicit detailed data on its development process. To structure, analyze, and enrich the data, we discussed it among the authors – involving an expert on the app's development (i.e., the second author acted as senior developer for the Corona-Warn-App) – to identify challenges and good practices. In more detail, we contribute the following:

- We report how and why the development process of the Corona-Warn-App differed from similar app-development projects at SAP SE (RQ_1 in Section 3).
- We describe practices that worked well during the development project to guide organizations in similar emergency situations (RQ_2 in Section 4).
- We discuss research opportunities for facilitating software development in emergency situations by reporting challenges that persisted through the development of the Corona-Warn-App (RQ_3 in Section 4).

Our case study contributes novel insights into the development of a COVID-19 contact-tracing app. While this exploratory case study is not full transferable to other situations and organizations, it sheds light into how the emergency development changed typical processes and the stakeholders involved. As such, our findings can guide other organizations in future emergency situations by avoiding potential pitfalls and reflecting on practices that worked well for developing the German Corona-Warn-App.

The remainder of this article is structured as follows: In Section 2, we describe our research methodology, including the general design of our case study as well as contextual background about the Corona-Warn-App—specifically, the involved organizations (Section 2.4) and the architecture of the app (Section 2.5). We report our insights on the app's development process in Section 3 and discuss challenges as well as practices in Section 4. Finally, we discuss threats to the validity of our study in Section 5, describe the related work in Section 6, and conclude this article in Section 7.

2. Case study design

In this section, we describe the design of our exploratory case study, for which we adapt the guidelines by Runeson and Höst (2009). Accordingly, we report the objective, theory, research questions, methods, and selection strategy we employed to conduct our study.

2.1. Objective

The Corona-Warn-App Germany was developed under exceptional circumstances as a means to handle the emergency situation of the COVID-19 pandemic. Consequently, the development process of this app exhibits unique properties that differ from the development processes used for other apps, for example:

- The app has been developed in an emergency situation that changed typical work styles and environments.
- The app had to be deployed as fast as possible to help manage the pandemic and mitigate health risks, which resulted in immense time and social pressure for the involved developers.
- The app is of high public interest, resulting in various stakeholders getting involved and the public critically observing its development.

Our objective with our exploratory case study was to understand how these unique properties impacted the development of the Corona-Warn-App, what challenges the developers faced, and what practices worked well during the project.

2.2. Theory

For our case study, we did not build on an established theory. Besides software-engineering research often lacking feasible theories (Runeson and Höst, 2009; Hannay et al., 2007), the development of the Corona-Warn-App in particular was a unique and novel case for which no theories could exist when we conducted our study. As a consequence, we decided to conduct an exploratory case study to describe what has happened and provide data for future research (e.g., for theory building). For this purpose, we used our point of view as outside observers who are familiar with software engineering and who aim to understand what could be useful to learn from this case for future emergencies.

2.3. Research questions

To address our objective, we defined three research questions (RQs) for our case study:

RQ_1 *How did the development process of the Corona-Warn-App deviate from the processes of other apps at SAP SE?*
First, we aimed to understand what the development process of the Corona-Warn-App looked like (e.g., agile versus traditional, involved stakeholders). Moreover, we compared this process to typical ones employed for app development at SAP SE to identify differences. Based on our insights, we identified which differences were caused by the unique properties of developing an app in, and for managing, the COVID-19 pandemic.

RQ_2 *Which of the deviations from the standard development process were helpful?*
Second, we aimed to identify which practices have been introduced due to the deviations in the development process. Particularly, we elicited practices that worked well and may prove beneficial in other emergency situations and organizations as well.

RQ_3 *Which of the deviations posed challenges?*
Finally, we aimed to identify deviations from the development process that resulted in notable challenges, and thus indicate opportunities for research to support developers in future emergency situations. We analyzed the root causes of these challenges and asked involved developers what support they would need to circumvent these.

Overall, our results provide a deeper understanding on how the German Corona-Warn-App has been developed (RQ_1); providing help for organizations in similar emergency situations (RQ_2) and guiding future research (RQ_3).

2.4. Case context: Organizations

Roughly 25 organizations as well as further individual developers have been involved in the development of the German Corona-Warn-App. In the following, we briefly describe the three primary stakeholders responsible for implementing and distributing the app: SAP SE, Deutsche Telekom AG, and the Robert Koch Institute (RKI). Furthermore, we briefly summarize how other stakeholders have been involved in the project.

SAP SE[1] is an internationally operating company and one of the largest vendors of Enterprise Resource Planning (ERP) software and services. SAP SE was founded in April 1972 by five former IBM employees in Walldorf, Germany. Currently, SAP SE has more than 100,000 employees around the world, delivering ERP applications that run on-premise and on the cloud, for instance, S/4HANA, SuccessFactor, Concur, and

---

[1] https://www.sap.com/about.html

Ariba. Regarding the Corona-Warn-App, SAP SE focused mainly on the actual software development, maintenance, and evolution.

Deutsche Telekom AG[2] is one of the largest telecommunication companies in the world, and operates in more than 50 countries. It is a Fortune 500 company located in Bonn, Germany, with more than 220,000 employees; focusing on internet and mobile telecommunication. For the Corona-Warn-App, Deutsche Telekom AG mainly provided the technical infrastructure (e.g., servers).

Robert Koch Institute (RKI)[3] is a German federal government research institute. The RKI employs roughly 1,100 employees (around 450 researchers) to monitor public health in Germany, inform the public, and advise the government. For instance, the RKI analyzes infection numbers and locations, and thus plays an essential role in Germany's fight against COVID-19. The RKI helped scope the Corona-Warn-App, define requirements, and test the app based on its medical expertise on the COVID-19 pandemic (e.g., defining distances for tracing contacts).

Other Stakeholders either contributed because of their governmental mandate (e.g., Office for Information Security, Ministry of Health), their own motivation (e.g., Chaos Computer Club,[4] researchers, individual developers), or requests from SAP SE and Deutsche Telekom AG (e.g., Fraunhofer Society, Google, Apple). For example, the Fraunhofer Society contributed practical knowledge on how to implement contact tracing via Bluetooth, allowing SAP SE to calibrate the Corona-Warn-App to match the requirements defined by the RKI. The Chaos Computer Club and various researchers performed extensive tests (particularly regarding security and data protection) and opened issues that were resolved by SAP SE or Deutsche Telekom AG to improve the app. Moreover, Google and Apple extensively supported the developers, providing the required support for implementing the app's features on their respective devices.

Overall, the variety of technical experience from different domains with direct input from authorities and the community created a fruitful, active, and successful development environment. However, this situation has also been unique for all involved stakeholders, resulting in new practices and challenges. With our case study, we aimed to explore this situation at SAP SE in more detail, since SAP SE was the stakeholder responsible for the actual software development of the German Corona-Warn-App.

2.5. Case subject: Corona-Warn-App Germany

In April 2020, the German federal government commissioned SAP SE and Deutsche Telekom AG to develop a COVID-19 contact-tracing app. Based on this partnership, the Corona-Warn-App[5] has been developed. The project has been highly complex, due to various factors related to the emergency situation of the pandemic (cf. Section 3). Despite these challenging factors, the initial release of the Corona-Warn-App was achieved – as an emergency project – within 50 days from the initial request. So, the app was launched in mid June 2020.

Overall, the project was highly successful, as demonstrated by the usage numbers of the Corona-Warn-App.[6] For example, between March 06, 2021, and October 11, 2021, around 1.5 million warning notifications were sent to the app users, on average 7,083 notification per day. Note that these notifications are only those from users who opted-in to share this information. As of August 12, 2022, the total number of downloads of the Corona-Warn-App for both Android and iOS exceeded 46 million downloads, its users shared more than 212 million test results, and the app provided over 177 million warnings to its users.

In addition, the Corona-Warn-App did constantly evolve, for example, to allow users to manage vaccination certificates or inform them about the current status of the pandemic.

Contact Tracing. Managing the high number of users and test results while protecting privacy required a versatile architecture. To address security and privacy concerns regarding the data that contact-tracing apps gathered (e.g., locations, health data), three main architectures for matching contacts within such apps have emerged, i.e., centralized, decentralized, and hybrid (Ahmed et al., 2020). They essentially differ in how they generate and store device IDs:

Centralized: In a central architecture, a central server generates temporary identifiers (with an expiry date). These temporary IDs are exchanged between devices of users through the devices' wireless interfaces when they are in close proximity. Upon a SARS-CoV-2 infection, users could upload their encountered IDs and the server notifies other potentially infected users. A prominent implementation of this centralized architecture is the *Bluetrace* protocol (Bay et al., 2020).

Decentralized: Using a decentralized architecture, the interaction between devices and servers is reduced to a minimum. Identifiers are generated locally on each user's device, not on a central server. So, upon a SARS-CoV-2 infection, only the infected user's identifiers are transferred to the server, the contact identifiers remain local on the device (Azad et al., 2020). Other users frequently retrieve the newest identifiers of infected users, which are matched against the locally stored contacts. A representative protocol for this implementation is the Private Automated Contact Tracing (PACT) protocol (Rivest et al., 2020), and Google's as well as Apple's Exposure Notification Frameworks (ENFs) follow a decentralized architecture, too (Vukolic, 2020).

Hybrid: A hybrid architecture divides tasks between the user's device and a trusted, centralized server. Hybrid architectures manage the temporary identifiers of devices locally, but shift the risk and user-notification management to the server. The server obtains information about the risk computation and exposure statistics, but preserves the anonymity of the devices and their users (Ahmed et al., 2020).

For the German Corona-Warn-App, a decentralized architecture was used to fulfill German security and privacy regulations as well as to comply with various stakeholder requirements (e.g., concerns of the public). The importance of security concerns is also reflected in the team setup for developing the Corona-Warn-App, which involved five sub-teams responsible for (i) iOS, (ii) Android, (iii) back-end, (iv) security and data privacy, as well as (v) testing.

Architecture. In Fig. 1, we display an overview of the different components of the Corona-Warn-App.[7] The Corona-Warn-App uses Google's and Apple's ENFs for Android[8] and iOS,[9] respectively. These frameworks generate temporary identifiers, called Temporary Exposure Keys (TEKs), for the devices and store them locally. From the TEKs, Rolling Proximity Identifiers (PRIs) are derived and exchanged through Bluetooth Low Energy. If a user has been diagnosed with SARS-CoV-2, they can decide to upload their TEKs (in this context called diagnosis keys) to the Corona-Warn-App server. To prevent misuse, the validity of the test must be ensured, either through transmitting a digital test result or by involving an   authorization code in the form of a
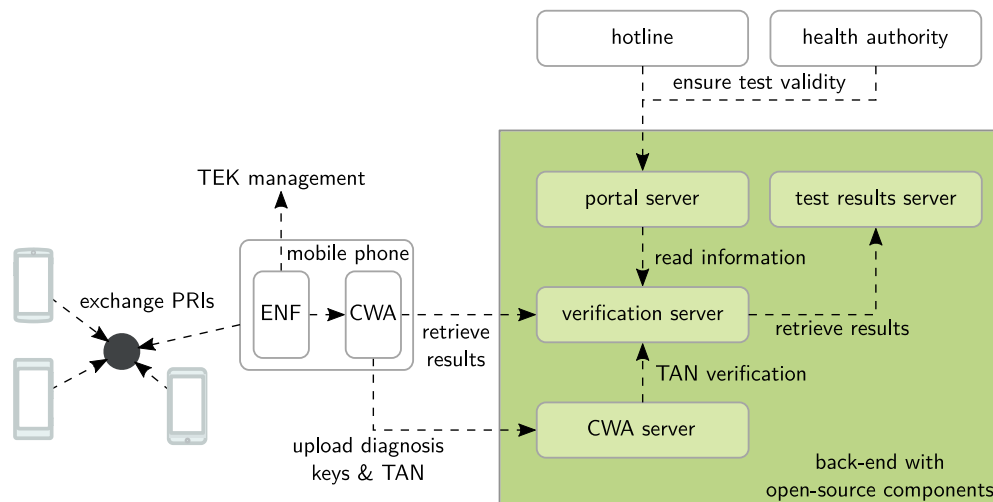

Fig. 1. Architecture of the Corona-Warn-App.

transaction number (TAN). The authenticity of this proof is ensured by the verification server, which enables the upload to the Corona-Warn-App server. Afterwards, the Corona-Warn-App server makes the keys of that device available for all Corona-Warn-Apps to download. On all devices running the Corona-Warn-App, the downloaded key packages are provided to the locally running ENF. The ENF identifies, matches, and provides epidemiologically relevant data of encounters to the app. In turn, the Corona-Warn-App is able to compute the risk level of its users and provide the respective information to them.

2.6. Data collection and analysis

At SAP SE, we had access to three data sources for collecting information on the development process of the Corona-Warn-App: documentation, interviews, and expert input. We decided to rely on all three to complement the individual strengths and weaknesses of the sources. In the following, we detail how we collected and analyzed information from each source.

2.6.1. Documentation

To design our interview guide and put our interviewees' answers into context, we also relied on the official documentation of the Corona-Warn-App[7] as well as repository data. Precisely, the first author elicited

information on the app's architecture (cf. Section 2.5) and development process. For this purpose, he first read through the official documentation to familiarize himself with the Corona-Warn-App and its development practices (e.g., how to submit bug reports). The first author noted down software-engineering related topics to structure the content of the documentation, thereby creating a reference sheet on where to find information on the specifics of developing the Corona-Warn-App. Most helpful was the description of the app's architecture, which provided an overarching understanding of the components and a common terminology we could use for the interviews. To refine his understanding and clarify any uncertainties, the first author further discussed independently with developers involved in the Corona-Warn-App. Moreover, he mined the version history of the app, for instance, to understand the extent of changes implemented over time and the novelties provided in releases. Based on these insights, we collaboratively drafted a first version of the development process. Then, we identified potential deviations from other projects based on the understanding of the first and fifth author—who are actively involved in developing apps at SAP SE. Lastly, we refined the process based on the interviews and the expert's knowledge, leading to the processes we display in Fig. 2 (overall Scrum process), Fig. 3 (process of incorporating new features) and Fig. 4 (detailed implementation process).
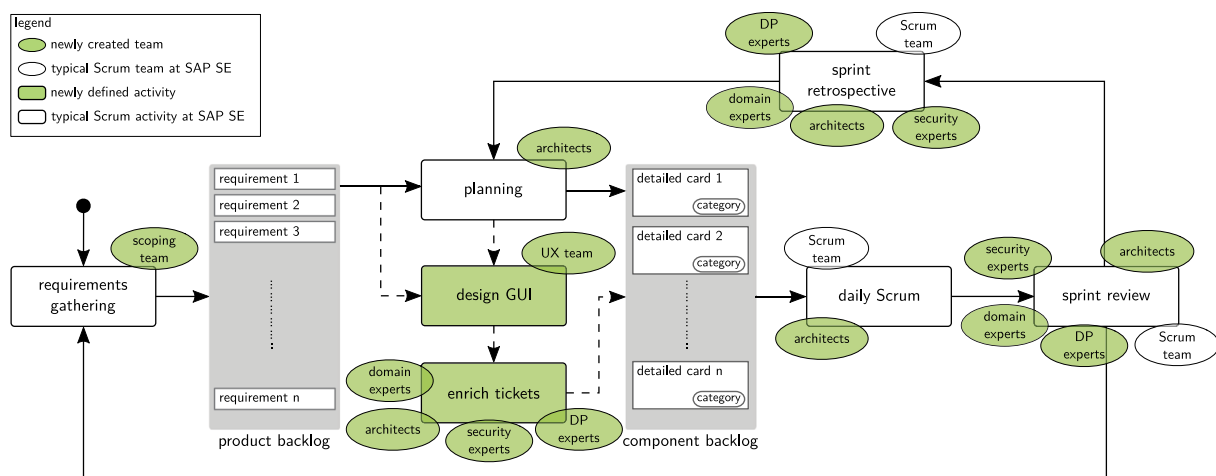

Fig. 2. Scrum process with adaptations for developing the Corona-Warn-App highlighted in light green and with dashed arrows. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Overview of our semi-structured interview guide.

ID	Question
Section: Planning	
P$_1$	How did/does the team formation affect the planning phase (i.e., involvement of different stakeholders)?
P$_2$	There are several inputs for defining features (e.g., requirements, community, legal, . . .). How did/does the team manage these inputs and translate them into backlog items/features?
P$_3$	Has the Corona-Warn-App planning phase changed since the beginning of the development or after the first release (i.e., maintenance)? If so, why?
Section: Implementation	
I$_1$	What are the differences between a normal Scrum process and the development process of the Corona-Warn-App?
I$_{1.1}$	Why were these changes needed?
I$_{1.2}$	What have been the consequent benefits and drawbacks?
I$_2$	Since the Corona-Warn-App is an open-source project:
I$_{2.1}$	How did/does the community support its implementation?
I$_{2.2}$	What have been problems and benefits of involving the community?
I$_3$	How was/is the communication between SAP SE, Deutsche Telekom AG, the open-source community, and other stakeholders organized?
I$_4$	Did the Corona-Warn-App implementation phase change since the beginning of the development or after the first release (i.e., maintenance)? If so, why?
Section: Testing, Quality Assurance, and Security	
T$_1$	How was/is the Corona-Warn-App tested?
T$_{1.1}$	How did/do teams coordinate to ensure the test coverage of features?
T$_{1.2}$	For integration tests, how did/do teams coordinate to locate issues?
T$_2$	What were/are the main sources for bug discovery? Particularly, how did/does the community help in this regard?
T$_3$	Security and data privacy are major concerns for the Corona-Warn-App:
T$_{3.1}$	Were additional steps introduced to the Scrum process to check security/privacy concerns?
T$_{3.2}$	Was there a parallel process focusing on security/privacy concerns?

2.6.2. Interviews

From our analysis of the documentation and the exploratory discussions with involved developers, we noticed that the development process of the Corona-Warn-App deviated from typical development

Fig. 3. Process to add features into the product backlog.

processes at SAP SE, due to several unique properties (cf. Section 3). To obtain a deeper understanding of the development process, we decided to conduct interviews with different members of the respective project. For this purpose, we designed a semi-structured interview guide based on our research questions as follows.

Design. To design our interview questions, we built on our experiences with this research method, reflected on our exploratory discussions, and consulted existing guidelines (Runeson and Höst, 2009; Glasow, 2005). More specifically, the first author started to draft interview questions based on the exploratory discussions and his expertise as an SAP SE developer with a detailed understanding of the development cycles at SAP SE. Then, the first and last three authors collaboratively reviewed and revised the interview questions, taking the perspectives of external researchers (third, fourth, and last author) as well as a team lead (not of the Corona-Warn-App) at SAP SE (last author). Note that the second author did not participate in these steps, due to his special role as an expert on the development of the Corona-Warn-App (cf. Section 2.6.3). After we revised the interview questions, we ran a pilot interview with the expert to test its comprehensibility and fit to answer our research questions. The pilot study did not yield any necessary adaptations to the interview questions. Consequently, we were confident that we covered the relevant aspects comprehensibly and could continue with the actual interviews.

Interview Guide. Depending on the role of an interviewee, we intended to ask different questions, which we display in Table 1. In the first section, we focused on identifying and understanding differences in the **planning** of the project. This section included the team composition (P$_1$), analysis of stakeholder inputs (P$_2$), and a general question on changes in the planning (P$_3$). In the second section, we were concerned with the actual **implementation** of the Corona-Warn-App. With these
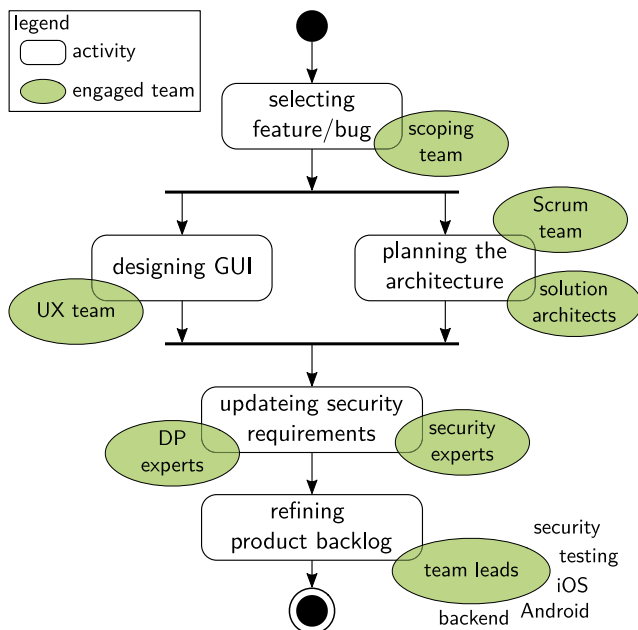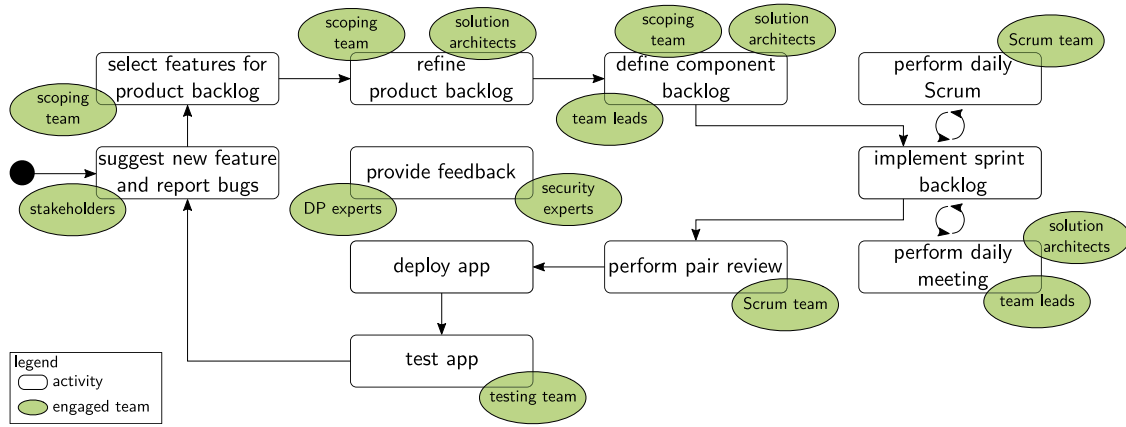
Fig. 4. Development process of the Corona-Warn-App.

Table 2
Overview of our interviews.

Interviewees	Duration	Date	Questions
Senior Developer	60 min	01/03/2021	I
Software Developer	60 min	24/03/2021	I
Software Developer	60 min	24/03/2021	I
Communication Manager	30 min	10/05/2021	I, T
Solution Architect	60 min	15/05/2021	P, I
Quality Assurance Lead	60 min	09/09/2021	T

questions, we focused on differences to typical development processes at SAP SE (I_1), the adoption of open-source (I_2), communication between stakeholders (I_3), and again a general question on changes regarding the implementation of the app (I_4). Note that we compared the development process to other Scrum projects (I_1), since the Corona-Warn-App has been developed using Scrum. In the last section, we were concerned with **testing and quality assurance**, particularly regarding the app's **security**. We elicited data on the general testing strategy (T_1), sources for discovering bugs (T_2), and how security or privacy concerns for the Corona-Warn-App changed this phase in general.

Conduct. We asked team members of the Corona-Warn-App to participate in our interviews, with candidates being proposed by the second and last author during our discussions. The participation was completely voluntarily, could be stopped at any point by the interviewee, and we did collect a minimum of personal information to ensure the interviewees' anonymity. Prior to each interview, we shared our interview guide with the interviewees. Then, the first author conducted a recorded interview and discussed our questions with the interviewee. Note that we used the questions only as a guide, but allowed rephrasing in case it would improve the interviewee's understanding.

We conducted six interviews from April to September 2021. In Table 2, we provide a summary of the interviews. As we display, the interviews typically took one hour, with only one shorter interview of 30 min. Since we invited interviewees with different roles—aiming for diversity over similarity, as recommended for qualitative surveys (Wohlin et al., 2012), it was not useful to ask all of our questions to each interviewee (i.e., we received most responses for the implementation phase). Still, as we show in the last column of Table 2, we conducted a detailed discussion about each section of our guide in at least one interview. For instance, the Quality Assurance Lead contributed the most detailed insights into the testing strategies employed for the Corona-Warn-App development. Due to the small size of the Corona-Warn-App team, we could not conduct many more interviews, but because we used these interviews to enrich the expert's (cf. Section 2.6.3) knowledge with complementary insights only, we argue that this does not threaten our case study. Note that we focused on the development process itself, which is why stakeholders from outside of the core development team of the Corona-Warn-App at SAP SE cannot provide additional insights for answering our research questions.

2.6.3. Expert knowledge

During our case study, we discussed intermediate results and potential adaptations to our analysis among the authors of this article, involving experts from SAP SE. Most importantly, we have been supported by an expert who has a detailed understanding of the app's overall development (second author), since he served as the senior developer in the project and contributed extensively to the development. After we analyzed our data, we derived and discussed our main insights among all other authors (i.e., an independent analysis without the expert). Then, the second author (i.e., the expert) provided additional insights and clarified misunderstandings or vague details. For instance, the second author stated that "the very early involvement of domain experts, for example, in the fields of security and privacy, ensured that the system architecture could be determined before starting the actual development." He further underlined the importance of constant communication among all involved stakeholders and the formal documentation of proposals and architectural decisions. Precisely, the second author remarked that "through constantly updated architecture documentation, misunderstandings can be effectively prevented and time be saved in the process." Finally, he emphasized that for the project to run efficiently, this communication needed to work across team as well as organizational borders. Namely, the successful cooperation between the different organizations was achieved by "direct communication channels, regardless of the department or company somebody works in." Such detailed insights were a tremendous help to understand the development of the Corona-Warn-App to structure, enrich, and refine our data.

3. RQ₁: Development process

From our interviews, we identified six primary factors (F) that increased the complexity of developing the Corona-Warn-App compared to other projects at SAP SE:

- F_1 The small period of time in which the app had to be developed to address the pandemic emergency.
- F_2 The new working models that the pandemic mandated, most prominently home office.
- F_3 The different organizations that were involved in the project, such as SAP SE, Deutsche Telekom AG, RKI, and the German Ministry of Health.
- F_4 The high privacy and security standards that the Corona-Warn-App had to fulfill.

F_5 The open-source nature of the app, which raised challenges regarding developers' ability to participate in and review the development process of the app.

F_6 The huge user base that the app targeted from the beginning (i.e., the whole German population that uses various devices from many vendors with different software as well as hardware components).

In the following, we elaborate on how these factors impacted the development process and, in Section 4, how these connect to the practices employed as well as challenges faced. For this purpose, we reference the labels of the respective factor in the following.

By synthesizing from all our data sources, we found that the Corona-Warn-App team, not surprisingly, faced several challenges during the planning phase of the project. For example, the involved team members had to form appropriate sub-teams, enable their collaboration (F_3), and ensure that they could deliver a secure as well as high-quality app within the limited period of time (F_1); all while working in the emergency situation caused by the pandemic (F_2). To overcome such challenges, the team members adapted their usual development processes to fit the specific circumstances of their situation. In Fig. 2, we illustrate a general Scrum process, which is typically employed to develop software at SAP SE—and in similar forms in many other organizations. The activities and roles highlighted in light green in Fig. 2 distinguish the adaptations the Corona-Warn-App team added to cope with the specifics of developing that app. Identically, the dashed arrows represent the workflows involving the newly introduced activities.

A specifically introduced scoping team created the product backlog and scoped the project by gathering requirements in meetings with the different stakeholders (cf. Sections 2.4 and 2.5). This scoping team involved requirement engineers and product owners (but no developers) who talked to the customers and decided what should be added to the app in which release. Mainly, the team derived requirements based on the stakeholders' input or their own expertise. To validate the requirements, the scoping team consulted with the solution architects team regarding the technical feasibility and experts (e.g., from the RKI or Fraunhofer Society) to align with epidemiological knowledge. After that, the scoping team worked with the development team to break the requirements down into concrete specifications and backlog items. This change has been implemented to facilitate the continuous communication with the various stakeholders involved (F_3), and to assure that the requirement analysis is transferred correctly to the development team. As a consequence, the resulting requirements were more robust and the expectations of the stakeholders well-defined. As for typical Scrum processes, the output of this step is the overall product backlog.

Before the actual implementation started, the solution architects team defined which components were required and through which interfaces they would interact. During a planning meeting, the team determined which items should be transferred from the product to the components backlog (which then represented the sprint backlog). In parallel, new features were directly communicated to the user-experience (UX) team, which was responsible for designing the graphical user interface (GUI) and creating respective specifications. The UX team prepared mock-ups of the user interface that could be validated with the stakeholders before the actual developers committed to implementing items from the backlog. This deviation helped SAP SE when discussing the design and implementation of the Corona-Warn-App with external stakeholders. In turn, the developers could directly start on more robust requirements, which saved time 'during the actual development.

Based on the mock-ups, security experts, data-privacy (DP) experts, and the architects worked with domain experts to add their specific requirements to individual feature tickets (F_3, F_4). Particularly, this group of stakeholders determined how to handle security and DP requirements, as well as how to implement epidemiological requirements. This

step was introduced to cope with the highly specific medical and data-protection requirements, which constantly evolved with new scientific findings on the COVID-19 pandemic and changing legal regulations. So, the process of transferring items from the product backlog to the component backlog became more complex, but helped ensure that the various stakeholder requirements would be considered.

Next, following the typical Scrum process, the product owners and architects of each component of the Corona-Warn-App derived concrete sprint-backlog items. There have been two types of daily Scrums (cf. Fig. 4): First, involving the Scrum team of the respective component only. Second, involving that Scrum team and representatives of all components as well as the solution architects of the Corona-Warn-App to keep the whole team updated about the progress of the overall project. During the typical sprint reviews and retrospectives, solution architects, domain experts, security and DP experts, as well as the development team ensured that all requirements were adhered to. Involving these experts in the reviews and retrospectives reduced the risk that the implementation of the Corona-Warn-App would require adaptations later on.

Development Process (RQ_1)

> *The development of the Corona-Warn-App followed a Scrum-like process, involving several modifications to integrate stakeholders from different organizations and domains. In particular, additional experts and activities for handling health as well as security concerns more directly were involved.*

---

4. RQ$_{2\ \&\ 3}$: Practices & challenges

Next, we discuss the deviations we identified in the development process in more detail. Particularly, we report on the practices the Corona-Warn-App team implemented to guide organizations in similar situations (RQ_2), and the challenges the team faced to guide future research RQ_3. Note that while facing several challenges when developing the app, the Corona-Warn-App developers figured out solutions for most challenges and delivered the project in a timely manner. We hope that presenting how the team resolved challenges will help other organizations in developing solutions for managing future emergency situations. While these solutions may not be suitable in every situation, they can help building a foundation for defining standardized practices and guidelines, understanding the impact of a solution, as well as developing new techniques for supporting developers that would be immensely helpful for future emergencies.

4.1. Team formation

A first challenge of developing the Corona-Warn-App was to set up a fitting team and enable the collaboration between organizations in a pandemic with severe restrictions regarding traveling and working in the office (F_2, F_3).

Solution-Architects Team. In contrast to typical development projects at SAP SE, developing the Corona-Warn-App has been an exceptional and time-critical request by the German government (F_1). When SAP SE and Deutsche Telekom AG started working on the project, they formed the solution architects team. The main task of this team was to derive the architectural blueprints for the app based on the available technology (particularly drafting the use of the ENF for a decentralized architecture), the technical requirements (e.g., bandwidth estimations), and the required usage patterns (e.g., the requirement to only allow verified tests to be used for a warning). Those decisions played an essential role in forming the sub-teams and involving the experts needed to make the project a success. Concretely, one interviewee stated that they had to know the technical blueprints to involve the right stakeholders in the project:

> "*After we had found out what we need as a development resource, we started recruiting people for the project.*"

The practice of first defining the requirements and involving corresponding experts worked well, even though the emergency situation may prompt developers to simply start implementing a solution. This would easily have lost important time, since the requirements for the Corona-Warn-App were highly specific (e.g., health data, security) and changed regularly (e.g., due to new research findings or policies), which required constant monitoring by experts.

Cross-Organization Collaboration. SAP SE and Deutsche Telekom are two large companies with different processes, environments, and cultures. Consequently, it was a challenge to form a larger team from these two companies and enable them to work in harmony in a short timeframe. The employees of both companies knew the positive impact of the project on the public, which lifted the teams' motivation to collaborate and coordinate to successfully deliver the Corona-Warn-App. Furthermore, the managers and architects made sure to dismiss the boundaries between the two companies by encouraging the teams to stay connected and communicate directly. As one interviewee expressed:

> "*We are here to solve a problem. I had a great communication experience, if there is an issue we call the involved person directly regardless of the imaginary boundary of company or location.*"

To enable this communication, the project management defined the responsibilities of each team and team member clearly, so that the team members knew whom to contact for what issue. This solution worked well according to the feedback provided by our interviewees, with one stating:

> "*If there is a question, I directly contact the responsible person without thinking about which organization [they] work in.*"

It has been a good practice to define clear roles and responsibilities within the development team, even beyond organization boundaries. This practice also removed complicated communication steps in such cross-organization processes. As a consequence, the involved developers could communicate with experts faster and more directly, as required to tackle an emergency situation.

Practices (RQ₂): Team Formation

Despite being in an emergency situation, developers should not simply start implementing a software system. Instead, we advise to first identify the most important requirements and corresponding experts to set up a managing team with clear roles and responsibilities. Developers should be able to directly communicate with the experts, ignoring organizational boundaries, to resolve problems faster and to build trust.

Research Challenges (RQ₃): Team Formation

To help in future emergencies, researchers should aim to improve the support for initiating, on-boarding developers to, and communicating in cross-organization teams, using empirical studies to test practices and corresponding techniques.

4.2. Planning

Planning new features and releases of the Corona-Warn-App posed several challenges to the development team, primarily because of sudden changes in regulations and in requirements related to health aspects as well as data privacy, which increased the time pressure even further (F_1, F_4).

Gathering Requirements. The features that should be developed in a sprint were determined during the planning phase of the development process (cf. Fig. 2). However, the Corona-Warn-App consists of several components that should deliver independent, yet consistent, features, for instance, for contact tracing and certificate management. Each component has its own architect and backlog, which must be synchronized to achieve the overall goals of the Corona-Warn-App. In the project, the scoping team was responsible for identifying the next set of features that the team should deliver. Then, the architects enriched the requirements with technical details that were required for the implementation. Finally, the team lead of each component translated the technical requirements and specifications into backlog items for the development team.

Determining the functional and non-functional requirements of a new system is a critical step in software engineering, since even a slight misunderstanding between the engineers and the stakeholders wastes time and money (Silhavy et al., 2011). Wasting time is not acceptable in time-critical development projects that tackle an urgent emergency, such as the Corona-Warn-App. Therefore, the Corona-Warn-App project introduced the scoping team to regularly meet with stakeholders to discuss and analyze new or changing requirements. Then, the scoping team mapped each requirement to a specific release of the app. In addition, the team met with security and DP experts to receive their feedback and requirements with respect to each new feature. So, the scoping team was, as defined in an interview,

> "*a separate team which decides what needs to go into the application, aligns the features with the customer and stakeholders, then maps these features to different releases.*"

Having an independent scoping team that involves and communicates with all relevant stakeholders (e.g., domain experts, developers, or security and DP experts) has been perceived as a good practice to manage the development of the Corona-Warn-App. Particularly, the team helped handle the complex situation and time criticality of the project.

Adding Features. In Fig. 3, we illustrate the process for adding a feature (or initiating a bug fix) to the backlog of the Corona-Warn-App. This process started with the scoping team selecting which features should be developed in the next release. The outcome of this process were refined backlog items that served as input for the team leads to create their respective component backlogs.

After selecting the new features for the next release, these were shared with the UX team and the solution architects (cf. Section 3). The UX team checked the new features regarding whether changes in the GUI of the Corona-Warn-App were required. If so, the UX team created a mock-up of the new GUI. This mock-up was used in further discussions with the stakeholders, and was validated by the stakeholders before the respective development sprint starts.

In parallel, the solution architects team determined what changes to the app's architecture and communication channels were required. Moreover, the team identified which components were impacted by the new requirements. As mentioned by one of our interviewees:

> "*Whenever a big problem shows up or a new requirement comes, the*"

> *solution architects analyze it from a holistic perspective to identify where the problem comes from. Then, [they] involve the right people in the discussions."*

So, at this point, the development team got involved and defined the end-to-end data flow for each new functional requirement. If needed, the architects met with stakeholders and medical or epidemiological experts to validate the proposed solution against their requirements and to confirm the epidemiological reasoning. Then, the solution architects team enriched the new requirements with technical details needed for the development teams. Afterwards, the solution architects team coordinated with security and DP experts to inspect the architectural changes and proposed solutions for the intended features. As a result, the appropriate changes were incorporated into the security and DP requirements.

Finally, the output of the previous steps was documented in a ticketing system. This documentation was used by the team leads to create detailed backlog items for the development teams, which now had the information required to implement the features. Notably, this process requires immense flexibility to tackle urgent requirement changes (e.g., due to new research findings) or severe bugs. In such cases, the process may have not been fully executed, and the architects rapidly gathered people to define backlog items:

> *"It is not a fixed process, sometimes we might face unspecific and urgent challenges, so we just need to get the right people together and keep going."*

While a structured process helps developers coordinate and guide the overall development, it has been an invaluable practice for the Corona-Warn-App to have people that can rapidly gather a team to tackle an urgent problem.

Practices (RQ₂): Planning

In a novel emergency situation, it is important to set up a team with experts on the corresponding requirements that can directly contact all involved stakeholders, helping to define the most important features and accelerating the development. While a defined process for adding new features or bug fixes helps structure the development, an emergency situation will cause constant deviation to handle urgent requests, which should be directly managed by the expert team.

Research Challenges (RQ₃): Planning

To help in future emergencies, researchers should design techniques and processes that allow to flexibly integrate different stakeholders at any point in time. Agile processes are helpful in this regard, but better support for rapidly specifying, documenting, and integrating requirements of different stakeholders (e.g., epidemiologists) is needed.

4.3. Implementation

The Corona-Warn-App consists of several components (cf. Section 2.5) that were implemented and maintained by different teams (F_3, F_4). As we described in Section 4.2, the team leads worked with the solution architects to create detailed backlog items for the development team. Each backlog item contained all pieces of information needed to implement the defined features, including details on: (i) the GUI design, (ii) communication interfaces, (iii) technical requirements, (iv) stakeholder requirements, (v) security requirements, and (vi) DP requirements. After defining this information, the actual implementation of requested features started. We display a detailed overview of this development process for the Corona-Warn-App in Fig. 4.

Programming. Based on the backlog items defined for the next sprint, a development teams' backlog items for each component were defined in collaboration between its architects and product owner. While implementing the features, each team had its own daily Scrum to report progress and ask for help if needed. Likewise, the solution architects and the teams' architects had a daily meeting to make sure the development of all components were aligned, and to clarify any occurring issue. In addition to the daily meetings, the solution architects had a weekly meeting with the stakeholders to keep them updated and resolve any remaining issue.

Reviews. As an explicit step, the Corona-Warn-App developers introduced a pair-review into their development process. In this step, after a developer finished working on a feature in a feature branch, the developer opened a pull request. Afterwards, the pull request was reviewed as soon as possible by the authoring developer and a senior developer (i.e., the pair) to reduce the time needed for the code to reach the development branch.

Deployment. Once a new version of any component was deployed (after passing automated tests that are integrated in the build pipeline) the test team started to test the app. In the end, the team submitted a testing report to the architects. As a result, the architects could determine the best fitting team member for any bug found and guided them in resolving the bug. During all steps of the process, the security and DP experts were involved and available to provide feedback for the Corona-Warn-App developers regarding security.

Engaging the different stakeholders (e.g., GUI team, epidemiologists) throughout the development process helped the solution architects manage the implementation of the Corona-Warn-App more efficiently. Involving these stakeholders directly into the process based on constant feedback has been a good practice to reduce the time and amount of rework needed to develop the app. Particularly, the direct interaction between developers and domain, security, or DP experts, saved time when implementing new features.

Practices (RQ₂): Implementation

In emergency situations, we recommend to closely integrate the required experts into the development to save time when designing, implementing, and testing a system. Enabling close collaboration between developers and these experts helps implement features faster and with less rework.

Research Challenges (RQ₃): Implementation

To help in future emergencies, it would be immensely helpful for experts from various domains to provide continuous feedback during the development (and Scrum) process, for which we need techniques that support their understanding of the system and for tracing as well as visualizing information.

4.4. Testing

Any software should be tested using a well-defined testing strategy that is integrated into the development process. For the Corona-Warn-App this posed particular challenges (F_1, F_4, F_6), since it had to be developed as fast as possible for a highly heterogeneous user base (e.g., various devices and vendors) and for partially rapidly changing requirements (e.g., based on changing laws and knowledge about the virus). In parallel, the Corona-Warn-App was under constant critical observation, and testing the app was a key activity to improve public trust in its usefulness.

Testing Strategy. Since the costs of tracing and fixing a bug increase with the delay of its discovery, the testing team of the Corona-Warn-App was directly involved in each sprint. After the development teams

decided which features should be developed within the next sprint, the test team received a technical description document. This document contained information about the components that will be changed, the new features, and the expected behavior of each feature. As a result, the test team started with developing a new test plan that contained the new test cases and the required regression tests. If feasible, the test team collaborated with an automation team to create automated tests based on the test plan. Tests that could not be automated were applied manually by the test team. To validate the test plans themselves and stay informed about any changes of any components of the Corona-Warn-App, the test team had initiated their own weekly meetings with architects and senior developers. From our interviews, we learned that these practices enabled the testing team to define, validate, and set up their testing strategy more efficiently.

Bug Reporting. When the test team discovered a bug or unexpected behavior, the team opened a new ticket about it. The ticket contained all information needed to reproduce the bug, such as the operating system, type of device, screenshots, and audit logs. In addition to that, whenever possible, the test team debugged the bug and instructed the development team with respect to the specific component that the test team expected to be the origin of the bug. This additional step accelerated the identification and fixing of bugs.

To help identify the team responsible for fixing the identified bug, the test team attached a predefined tag to the bug-report ticket. Using these tags, the ticketing system assigned tickets to the responsible team architect or scrum master automatically. One of these two could assign the ticket to the corresponding developer. In case of uncertainty about any issue identified, the test team contacted the responsible development team directly and discussed the issue. This collaboration and direct communication helped to avoid unnecessary bugs to be reported, and to add more context to valid ones. Also, it became more efficient to validate bug reports and their fixes as well as to clarify ambiguities arising between the teams. Our interviewees noted this for the Android version of the Corona-Warn-App in particular, for which many different parameters (e.g., version, device, manufacturer) were needed to reproduce a bug—not all of which were accessible to the developers at all times. For developing the Corona-Warn-App, the dedicated testing team and its close integration into the overall development process via regular meetings proved to be a well-working practice.

Practices (**RQ₂**): Testing

To assure the quality of an emergency system, we recommend to set up a dedicated testing team, even if the urgency of developing the app may pressure organizations in not doing so. Involving the testing team directly into meetings and providing them detailed specifications helps define independent tests, with the testers' knowledge and independence yielding high-quality bug reports and better assignments to the responsible developers.

Research Challenges (**RQ₃**): Testing

To help in future emergencies, further automation for bug-reporting would be helpful, for instance, to guide testers in designing tests and assigning bug reports correctly. Facilitating such tasks for testers could greatly accelerate the process or resolving bugs and addressing urgent inquiries.

4.5. Stakeholder involvement

As discussed, the Corona-Warn-App was heavily structured around close stakeholder involvement, aiming to accelerate the development and deliver the app as fast as possible (F₁, F₃, F₅, F₆). Still, involving all stakeholders, other communities (cf. Section 2.4), and the general public to build trust posed novel challenges for the developers.

Stakeholder Engagement. To enable stakeholders (e.g., general public, researchers) and the open-source community to participate in all stages of the development process, already the first repository of the Corona-Warn-App was made public. At the beginning of the development, the community actively posted new feature requests and opened pull requests implementing these features. Since the Corona-Warn-App is an emergency app, the features requested by the official customers (i.e., RKI and the German government) were most pressing and most important to be implemented as fast as possible. So, the teams at SAP SE and Deutsche Telekom AG focused more on implementing these features and rolling out updates more frequently. This led to less direct participation of the community through GitHub, since it took time for the Corona-Warn-App team to review submitted code. Still, the Corona-Warn-App team always had the community in mind and wanted to enable more direct participation. Unfortunately, a lack of official processes to integrate code of the community into the app, security requirements, and constraints of intellectual property resulted in most community contributions not becoming part of the app. As a result, the main contributions of other stakeholders were pull requests, submitted feedback, and opened feature requests or bug reports.

Enabling Community Contributions. The Corona-Warn-App had a highly active supporting community, as mentioned by an interviewee:

> *"The people were extremely enthusiastic about the Corona-Warn-App, they took a lot of their time to look at the app, provide feedback, and suggest improvements."*

To improve how the community could contribute to the Corona-Warn-App, a dedicated community-management team was introduced later on. This team crafted and updated the policy on how the community could contribute to the development of the Corona-Warn-App.

Specifically, the community continued to contribute via the established features of social-coding platforms (e.g., pull requests, issues), keeping a workflow that is well-known to developers. Also, the community-management team opened up the Corona-Warn-App to allow the community to Beta test any new release. The respective testers could then report issues that were reviewed by that team, which also communicated with the tester to clarify any missing details. As a consequence, over time, this team became responsible for refining community comments, bug reports, and feature requests in the core Corona-Warn-App development-team backlog as well as for providing feedback to the community about their contributions. Finally, the community-management team developed a dashboard to help them interact with the community quickly and efficiently, particularly to cope with the high participation and use of various GitHub repositories for the different app components. This practice of establishing a dedicated team that manages community contributions and involves interested stakeholders in the development worked well, since that team allowed the developers to focus on the most important features, identified valuable contributions, and built trust to the public via direct communication.

Practices (**RQ₂**): Stakeholder Involvement

To handle public interest, built trust, and manage community contributions, we recommend to involve all stakeholders into the development of an emergency system through a dedicated team. The input of the different stakeholders (e.g., bug reports) are invaluable to assure the quality and increase the use of the system.

┌─ Research Challenges (**RQ₃**): Stakeholder Involvement ─┐

To help in future emergencies, we see the need for more insights and guidelines on how to involve various stakeholders into software development, particularly regarding building trust, balancing requirements trade-offs, and managing intellectual property. More support for such concerns in established tools like GitHub would be of great value.

└──────────────────────────────────────┘

5. Threats to validity

In this section, we briefly summarize the most important internal and external threats to the validity of our case study.

5.1. Internal validity

Regarding the internal validity, our insights may be threatened because we heavily built on the second author's experiences as senior developer of the Corona-Warn-App. However, since we focused on the general processes, practices, and most critical challenges the Corona-Warn-App developers experienced, his experiences are invaluable for our case study. To mitigate this threat, we interviewed six developers involved in the Corona-Warn-App, elicited additional data from the app's documentation, and internal discussions. These further data sources improve our confidence in the practices and challenges we elicited, but the threat still remains. Still, this is a typical threat for exploratory case studies and experience reports that elicit such data (e.g., on developers' communication) from industrial practice.

Another possible threat is the extent of the questions asked. While adding more in-depth questions would increase the recorded information, we argue that it would not lead to further aspects mentioned by the interviewers, but rather to repeating stand points.

5.2. External validity

The most relevant threat to the external validity of our case study is that we report only on a single case in which an app was developed within and for managing an emergency situation. This app development was exposed to unique properties, due to the nature of the project: A collaboration of several organizations, involvement of various stakeholders as well as the open-source community, and pressure of the COVID-19 pandemic. So, although the practices we reported helped the Corona-Warn-App team, they may not be suited for other circumstances and projects. Furthermore, the processes at SAP are most likely not fully transferable to other organizations, but we argue that our findings remain useful for other organizations that apply Scrum. Although more research is needed to validate our best practices in other projects, we are positive that they can help organizations in similar emergency situations. Moreover, we report a substantial case study on the internals of developing the German Corona-Warn-App, which are rarely available for researchers to study. Thus, we argue that our exploratory case study is immensely valuable for researchers and practitioners to build theories and manage future emergency situations, which are likely to occur even more often (e.g., monkeypox virus, natural disasters due to climate change).

Another threat to our work is the limited number of interviews we could conduct, due to the small team size of the Corona-Warn-App core team. Since we were aware of this issue from the start, we consulted an expert to double-check the plausibility of the given statements and reviewed the official documentation of the Corona-Warn-App. So, we tried to mitigate this limitation and contribute reliable insights. Furthermore, the selection of interviewees may threaten our findings, due to the limited number of interviewees having insights into the security, DP, and testing teams. However, we argue that more interviews with members of such teams would provide only few new insights. First, we are concerned with the software-development practices, but many stakeholders in these teams were experts on other areas and not involved in the actual software development. Second, the expert would have pointed us to any stakeholders that could have contributed significantly different insights. Consequently, we argue that this threat is mitigated.

6. Related work

We are not aware of any study that reports on the overall development of a COVID-19 contact-tracing app in the same level of detail as we do. Some related work are the studies by Bano et al. (2020), who analyze to what extent contact-tracing apps fulfill their defined requirements; and by Sutcliffe et al. (2021), who investigate the impact of values on such apps' requirements. However, neither do these studies provide details on other development steps, nor do they elicit information on the development from involved stakeholders. More research has focused on analyzing different tracing apps from a technical or ethical point of view (Garousi and Cutting, 2021; Morley et al., 2020; Sun et al., 2021; Ahmed et al., 2020; Abuhammad et al., 2020; Gupta et al., 2021; Erikson, 2021; Liang, 2020). In this context, the study of Reelfs et al. (2020) is the closest one to our own, since the authors study the Corona-Warn-App Germany. Still, none of these works provides detailed insights into the actual development processes, good practices, or challenges. Some other studies have been concerned with the impact of the COVID-19 pandemic on software developers (Ralph et al., 2020; Russo et al., 2021; Silveira et al., 2021). These studies focus more on the impact of the pandemic on the individual, whereas we focus on its impact on the development process of an emergency app in this article. Consequently, our contributions advance the existing body-of-knowledge with a substantial and detailed case study on the development of a contact-tracing app.

Similar case studies and experiences have been shared regarding software development in the COVID-19 pandemic and other emergency situations, but mostly on other software systems than an actual contact-tracing app. Closes to our work is the experience report by May et al. (2024), who report on the development of a COVID-19 certificate-verification system in a startup. While there is some overlap, the different natures of the systems and organizations (large companies versus startup) involved also lead to differences and complementary insights to ours. Bombarda et al. (2022) shared their experiences of developing medical software systems in the pandemic within an international team. Not surprisingly, several of their and our lessons learned are closely related and similar, improving our confidence in our case study—which contributes additional evidence from another case. For example, both studies led to the lesson that coordination among the involved teams was key. Our case further underpins the suggestion by Bombarda et al. that open-source practices may be helpful for emergency projects. Similarly, both studies agree, among others, on the importance of defining clear responsibilities, having structured code reviews, defining requirements early, as well as architecting a system properly even in an emergency. Other studies underpin these experiences regarding the communication among distributed teams in emergency situations (Plotnick et al., 2008) and the integration of ethical values into development processes (Nussbaumer et al., 2023). Besides contributing additional evidence on such previous findings, we also shared new insights from a different and highly specific case in this article.

7. Conclusion

In this article, we reported an exploratory case study on the development of the German Corona-Warn-App at SAP SE. For this purpose, we conducted six interviews with developers at SAP SE, analyzed the available documentation, and discussed our findings with a senior developer of the app. In summary, we found that:

RQ₁ Six key factors complicated the development of the Corona-Warn-App and resulted in modifications to the typical Scrum process employed at SAP SE, particularly to include experts on relevant topics of interest (cf. Section 3): (1) limited time; (2) new working models; (3) multiple organizations being involved; (4) high privacy and security standards; (5) open-source nature; and (6) diverse user base.

RQ₂ We learned about five practices that the developers of the Corona-Warn-App considered helpful to develop the Corona-Warn-App (cf. Section 4):

    (1) Facilitate cross-organization communication to ease the direct interaction between the stakeholders involved in the whole project;

    (2) Have architects gain a good overview understanding of the project, enabling them to involve the experts needed to implement specific features—particularly for time-critical requests;

    (3) Involve relevant domain experts directly into the development process from the beginning to reduce the need for reworking parts of the app and improving its quality;

    (4) Implement a parallel testing process via a dedicated, independent testing team that constantly synchronizes with the developers to obtain high-quality bug reports and apps; and

    (5) Engage stakeholders and the (open-source) community to receive external, rapid feedback.

RQ₃ Corresponding to each practice, we identified challenges as opportunities for future work that can help in other emergency situations (cf. Section 4):

    (1) Improving the on-boarding and communication in cross-organization teams;

    (2) Designing techniques and processes that enable the flexible integration of stakeholders;

    (3) Conceptualizing how to provide and trace information in a format comprehensible for stakeholders with varying expertise;

    (4) Automating processes for bug reporting and code reviewing; and

    (5) Developing and testing recommendations for involving various stakeholders in an emergency project to guide them in understanding the trade-offs between requirements.

While some of our insights are similar to related work (cf. Section 6), we contributed a unique and highly interesting case. Thus, our article provides additional evidence as well as completely new findings. We hope that our insights help guide future research in preparing software engineering for future emergencies.

CRediT authorship contribution statement

Mohamad Fawaz Enaya: Conceptualization, Data curation, Investigation, Methodology, Resources, Visualization, Writing – original draft, Writing – review & editing. **Thomas Klingbeil:** Investigation, Resources, Writing – review & editing. **Jacob Krüger:** Methodology, Supervision, Visualization, Writing – original draft, Writing – review & editing. **David Broneske:** Conceptualization, Methodology, Supervision, Writing – original draft, Writing – review & editing. **Frank Feinbube:** Conceptualization, Methodology, Project administration, Resources, Writing – original draft, Writing – review & editing, Supervision. **Gunter Saake:** Conceptualization, Project administration, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Some data is publicly available through other institutions and linked in the paper. Details of the interviews are confidential.

Acknowledgments

References

Abuhammad, S., Khabour, O.F., Alzoubi, K.H., 2020. COVID-19 contact-tracing technology: Acceptability and ethical issues of use. Patient Prefer. Adherence 14, 1639–1647. http://dx.doi.org/10.2147/PPA.S276183.

Ahmed, N., Michelin, R.A., Xue, W., Ruj, S., Malaney, R., Kanhere, S.S., Seneviratne, A., Hu, W., Janicke, H., Jha, S.K., 2020. A survey of COVID-19 contact tracing apps. IEEE Access 8, 134577–134601. http://dx.doi.org/10.1109/ACCESS.2020.3010226.

Azad, M., Arshad, J., Kamal, S., Riaz, F., Abdullah, S., Imran, M., Ahmad, F., 2020. A first look at privacy analysis of COVID-19 contact tracing mobile applications. IEEE Internet Things J. 8, 15796–15806. http://dx.doi.org/10.1109/JIOT.2020.3024180.

Bano, M., Zowghi, D., Arora, C., 2020. Requirements, politics, or individualism: What drives the success of COVID-19 contact-tracing apps? IEEE Softw. 38 (1), 7–12. http://dx.doi.org/10.1109/MS.2020.3029311.

Bay, J., Kek, J., Tan, A., Hau, C.S., 2020. BlueTrace: A Privacy-Preserving Protocol for Community-Driven Contact Tracing Across Borders. Technical Report, Government Technology Agency-Singapore.

Bombarda, A., Bonfanti, S., Galbiati, C., Gargantini, A., Pelliccione, P., Riccobene, E., Wada, M., 2022. Guidelines for the development of a critical software under emergency. Inf. Softw. Technol.

Erikson, S., 2021. COVID-apps: Misdirecting public health attention in a pandemic. Global Policy 12, 97–100. http://dx.doi.org/10.1111/1758-5899.12888.

Garousi, V., Cutting, D., 2021. What do users think of the UK's three COVID-19 contact-tracing apps? a comparative analysis. Health Care Inform. 1–7. http://dx.doi.org/10.1136/bmjhci-2021-100320.

Glasow, P.A., 2005. Fundamentals of Survey Research Methodology. Technical Report MP 05W0000077, Mitre Corporation.

Gupta, R., Pandey, G., Chaudhary, P., Pal, S.K., 2021. Technological and analytical review of contact tracing apps for COVID-19 management. J. Locat. Based Serv. 1–40. http://dx.doi.org/10.1080/17489725.2021.1899319.

Hannay, J.E., Sjøberg, D.I., Dybå, T., 2007. A systematic review of theory use in software engineering experiments. IEEE Trans. Softw. Eng. 33 (2), 87–107. http://dx.doi.org/10.1109/TSE.2007.12.

Liang, F., 2020. COVID-19 and health code: How digital platforms tackle the pandemic in China. Social Media + Soc. 6 (3), 1–4. http://dx.doi.org/10.1177/2056305120947657.

May, R., Baron, N., Krüger, J., Leich, T., 2024. Pandemic startup software engineering: An experience report on the development of a COVID-19 certificate verification system. J. Syst. Softw.

Morley, J., Cowls, J., Taddeo, M., Floridi, L., 2020. Ethical guidelines for COVID-19 tracing apps. Nature 582 (7810), 29–31. http://dx.doi.org/10.1038/d41586-020-01578-0.

Munzert, S., Selb, P., Gohdes, A., Stoetzer, L.F., Lowe, W., 2021. Tracking and promoting the usage of a COVID-19 contact tracing app. Nat. Hum. Behav. 5 (2), 247–255. http://dx.doi.org/10.1038/s41562-020-01044-x.

Nussbaumer, A., Pope, A., Neville, K., 2023. A framework for applying ethics-by-design to decision support systems for emergency management. Inf. Syst. J. 33 (1).

Plotnick, L., Ocker, R., Hiltz, S.R., Rosson, M.B., 2008. Leadership roles and communication issues in partially distributed emergency response software development teams: A pilot study. In: Hawaii International Conference on System Sciences (HICSS). IEEE.

Ralph, P., Baltes, S., Adisaputri, G., Torkar, R., Kovalenko, V., Kalinowski, M., Novielli, N., Yoo, S., Devroey, X., Tan, X., Zhou, M., Turhan, B., Hoda, R., Hata, H., Robles, G., Milani Fard, A., Alkadhi, R., 2020. Pandemic programming: How COVID-19 affects software developers and how their organizations can help. Empir. Softw. Eng. 25, 1–35. http://dx.doi.org/10.1007/s10664-020-09875-y.

Reelfs, J.H., Hohlfeld, O., Poese, I., 2020. Corona-warn-app: Tracing the start of the official COVID-19 exposure notification app for Germany. Ratio 10 (5), 10–12. http://dx.doi.org/10.1145/3405837.3411378.

Rivest, R.L., Callas, J., Canetti, R., Esvelt, K., Gillmor, D.K., Kalai, Y.T., Lysyanskaya, A., Norige, A., Raskar, R., Shamir, A., Shen, E., Soibelman, I., Specter, M., Teague, V., Trachtenberg, A., Varia, M., Viera, M., Weitzner, D., Wilkinson, J., Zissman, M., 2020. The PACT Protocol Specification. Private Automated Contact Tracing Team. Technical Report, MIT.

Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. 14 (2), 131–164. http://dx. doi.org/10.1007/s10664-008-9102-8.

Russo, D., Hanel, P., Altnickel, S., van Berkel, N., 2021. The daily life of software engineers during the Covid-19 pandemic. In: International Conference on Software Engineering: Software Engineering in Practice Track. ICSE-SEIP, IEEE, pp. 364–373. http://dx.doi.org/10.1109/ICSE-SEIP52600.2021.00048.

Seto, E., Challa, P., Ware, P., 2021. Adoption of COVID-19 contact tracing apps: A balance between privacy and effectiveness. J. Med. Internet Res. 23 (3), http: //dx.doi.org/10.2196/25726.

Silhavy, R., Silhavy, P., Prokopová, Z., 2011. Requirements gathering methods in system engineering. In: International Conference on Automatic Control, Modelling & Simulation. In: ACMOS, WSEAS, pp. 106–110.

Silveira, P., Mannan, U.A., Almeida, E.S., Nagappan, N., Lo, D., Kochhar, P.S., Gao, C., Ahmed, I., 2021. A deep dive into the impact of COVID-19 on software development. IEEE Trans. Softw. Eng. http://dx.doi.org/10.1109/TSE.2021. 3088759.

Sun, R., Wang, W., Xue, M., Tyson, G., Camtepe, S., Ranasinghe, D.C., 2021. An empirical assessment of global COVID-19 contact tracing applications. In: International Conference on Software Engineering. ICSE, IEEE/ACM, pp. 1085–1097. http://dx.doi.org/10.1109/ICSE43902.2021.00101.

Sutcliffe, A., Sawyer, P., Liu, W., Bencomo, N., 2021. Investigating the potential impact of values on requirements and software engineering. In: International Conference on Software Engineering. ICSE, IEEE/ACM, pp. 39–47. http://dx.doi.org/10.1109/ ICSE-SEIS52602.2021.00013.

Vukolic, M., 2020. On the interoperability of decentralized exposure notification systems. arXiv:2006.13087.

White, L., van Basshuysen, P., 2021. Without a trace: Why did corona apps fail? J. Med. Ethics 1–4. http://dx.doi.org/10.1136/medethics-2020-107061.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., 2012. Experimentation in Software Engineering. Springer, http://dx.doi.org/10.1007/978-3-642-29044-2.

Wymant, C., Ferretti, L., Tsallis, D., Charalambides, M., Abeler-Dörner, L., Bonsall, D., Hinch, R., Kendall, M., Milsom, L., Ayres, M., et al., 2021. The epidemiological impact of the NHS COVID-19 app. Nature 594 (7863), 408–412. http://dx.doi.org/ 10.1038/s41586-021-03606-z.