Network and Communication Technologies; Vol. 4, No. 1; 2019
ISSN 1927-064X E-ISSN 1927-0658
Published by Canadian Center of Science and Education

Research Study: Towards a Collaborative Trust-Based Debugging Model for the Telecom Industry

Ayse Kok Arslan[1]

[1] Researcher/Silicon Valley, Kok Bogazici University, Istanbul, Turkey

Correspondence: Ayse Kok Arslan, Researcher/Silicon Valley, Kok Bogazici University, Istanbul, Turkey. E-mail: ayshe.kok@gmail.com

Received: June 18, 2019 Accepted: June 24, 2019 Online Published: June 27, 2019

doi:10.5539/nct.v4n1p36 URL: https://doi.org/10.5539/nct.v4n1p36

Abstract

This research presents the architecture of a collaborative troubleshooting platform to establish user trust by presenting practical scenarios of problem diagnosis using the technique of root-cause analysis in a high tech company in Silicon Valley. The study provides details of the implementation of debugging techniques within the field of telecommunication and demonstrates the feasibility of this problem-solving approach in a collaborative trouble-shooting platform to establish user trust.

Keywords: model, trust-based, telecom industry

1. Introduction

Today's IoT (Internet of Things) compromised of smartphones and laptop computers along with other networked technologies do not only provide powerful functionality but also complex issues in case of a dis-functional network. Troubleshooting such complex issues is not easy given the requirement for sophisticated diagnostic tools, not to mention the required user interfaces to debug the problem directly (Sundaresan, Grunenberger, Feamster, Papagiannaki, Levin, & Teixeira, 2013).

Among the popular diagnostic tools used most commonly to mitigate the troubleshooting process are network diagnostic software integrated into the operating system, traditional command line tools, and various third-party tools for making an accurate diagnosis (Kim, Singh, & Schulzrinne, 2011; Cui & Biersack, 2011). Yet, for some tools there may be a need to implement different tasks to keep track of network packages (Aggarwal, Bhagwan, De Carli, Padmanabhan, & Puttaswamy, 2011), which might require more CPU power than those existing on smaller devices.

This paper suggests a collaborative troubleshooting platform that leverages debugging practice of mainly messaging apps and telecommunication devices. Based on a real-life example in a high tech company in Silicon Valley, its underpinning theory is that in case of a device suffering from network problems regarding Network Address Translation (NAT) box or DNS resolution, it offloads related debugging tasks to other chatbots that have more diagnostic capabilities. When another chatbot with a diagnostic functionality is alerted on the problem report, the diagnostic process can be initiated to find out about the network issue. Otherwise, task- forwarding to a more relevant device occurs.

A common message format is implemented to make the communication across chatbots possible and to enable them get reports of network issues. In addition to this, a diagnostic chatbot can convey requests for probe to other chatbots to receive more information from other devices. To achieve this, two layers are designed: *logic layer* and *probe layer* to fulfill both diagnostic and probe functionalities. The troubleshooting platform being subject to this study is mainly used by messaging applications for Android devices. While native APIs are used by the Android application to utilize the *probe layer*, the logic layer makes use of a Web-based application technology (Dong & Dulay, 2011).

2. Related Work

Various research studies have been conducted regarding the field of troubleshooting. A Web session diagnostics tool was developed by Cui et al. (2014) based on the correlation between packet measurements concerning several machines. In addition to these scholars, a trouble-shooting tool based on a deep learning technique based on signature was designed by Aggarwal et al. (2012), and an argumentation-based algorithm was developed by Dong et al. (2010) for network diagnosis.

There are also some diagnostic systems with a focus on network problems and performance degradation (Rayanchu, Patro, & Banerjee, 2011). However, it would be wrong to say that most studies made a research on networked devices.

Based on these studies, it can be inferred that in case of a user complaint about the decreased network performance, various recorded metrics ranging from jitter to throughput, RTT and packet re-delivery could be used in order to investigate the related metric underpinning poor performance. Although this approach is based on cooperation among devices probing in real-time with pre-determined rules for diagnostics rather than inter-dependent metrics should be used. Therefore, according to this approach, there is no need for devices to record network states. Furthermore, for effective collaboration among devices, useful mechanisms such as forwarding, probing and discovering are recommended.

3. An Overview of Platform Architecture

The major purpose of the troubleshooting platform subject to this study is to make an inquiry on the network issues that exist in devices across the network. The diagnostic process includes making use of collaboration of nodes within the network and getting connected to others. The platform also provides an opportunity for both the engineering support team and chatbots to make an exchange of problem profiles along with results of diagnostic.

The company that has been subject to the case study provides 24-hours support for all days of the week for its external partner companies as well as its employees regarding its consumer communications and business messaging products. In general, the following user types are provided support with by the company's engineering support team:

- Partners contacting us through issue tracker tool and support aliases
- Employees contacting through issue tracker tool and support aliases
- User feedback consolidated by Feedback Engineering Team and escalated for further troubleshooting

A very high level scope of support can be described as follows:

- Pre-triage of issues (i.e, ensure tickets- bugs and feature requests) are actionable by Engineering and Product Management teams
- Troubleshoot, provide technical guidance / support & where possible, resolve issues
- Follow up till issue closure
- Onboarding users & partners onto the Communications ecosystem (i.e, different levels of technical support etc.)
- Handling escalations
- Prioritizing & bubbling top issues to the right Engineering / Product Management teams through reports & escalations

A. Device Types

The following roles exist regarding the collaboration on the trouble-shooting platform:

- *Client device*: This refers to the smart object having a network issue.
- *Forwarding device:* This device acts as a transmitter for transmitting requests on probes to the device which provided diagnostics.
- *Device for diagnosis:* This device provides support for making a diagnosis of the issue.

Depending on the computing power, each device within a network fulfills a variety of roles , by being attached to user interfaces. To give a specific example, while a laptop computer may act as a *diagnostic device* because of some feasibility reasons, the laptop may at the same time act as a *client device* due to other capabilities. A forwarding device might be required if no communication could occur between a *client device* and a *diagnostic device* because of some technological reasons.

B. Device Registration

Registration of attributes for the device occurs by having a connection to the network. Among these attributes, one could count state of mobility (e.g., smartphones and tablets) and network interfaces. The role of a *diagnostic device* is to maintain a device directory based on these profiles to make use of it in the case of problem diagnosis. For example, if there is a need for a 3G network those devices that can reach a 3G network would be looked for so that a probe request could be sent.

C. Problem Description and Forwarding

In case of a network problem detection, the following steps are performed:

- A device first provides a problem description including failure symptoms (no wireless connection).
- Next, a diagnosis request is made to the *diagnostic device*.
- In case a device cannot reach a *diagnostic device*, the diagnosis request fails.
- In such a case, recipient devices of the diagnosis request forward the request to other devices.

D. Mobile Devices

In case of a Wi-Fi issue, a user can pair the device via Bluetooth so that it is capable of transmitting the diagnosis request.

E. Diagnosis

There are predefined diagnostic rules to initiate the diagnostic process. The preliminary rule is to check whether there has been a report of the same problem by other devices. If the answer is no then there is a communication attempt with other devices to specify if those can be accessed without any network issues. In case the same issue happens for various devices in the network support team would infer the network infrastructure as the root cause and run other diagnostic software to see whether Wi-Fi or DNS, DHCP, TCP/UDP function correctly.

In case of a device failure connection to the central management server, the knowledge regarding the existence of the same issue on other devices or computers in different networks would be helpful. In addition to this knowledge, a confirmation of whether collaborative nodes in external networks are able to transmit network packets to the devices would ensure that incoming packets are received correctly.

4. Diagnosis Scenarios

This section explains various exemplary scenarios regarding the problem diagnosis process which takes place on the collaborative trouble-shooting platform.

A. Device Diagnostics using History

If there is a significant or higher priority issue, the diagnostic process is initiated by delivering requests on probe to other devices to specify if the same problem exists.

B. Smartphone Diagnostics

If a smartphone application might have a network issue, push notifications are used to deliver network packets to mobile devices for many Android device applications. To determine if these notifications function correctly, probe requests are sent to any intermediate communication methods available via Bluetooth.

D. Computer Diagnostics

In the case of a temporary ISP outage and no Internet connection, the first discovery would be for a device within the network. Next, a diagnosis request would be sent through means of a 3G network.

5. Making Use of Root Cause Analysis

A root cause analysis process start by asking probing questions that will make symptoms distinctive from actual problems/opportunities. After the symptom specification, the question 'Why' will specify other symptoms or the underpinning root cause to be identified. Once the root cause identified, engineering and support team start to recommend general objectives, measurable performance criteria, and constraints. A sample analysis sheet has been included in Appendix A. Also, some best practices for troubleshooting scenarios have been included in Appendix B.

By means of root cause analysis, actual problems, symptoms and opportunities are made clear so that engineering support team can recognize if the root cause relates to a technical issue and if so, resolution process can start by making use of the systems process approach.

A. Creating a Root Cause Analysis Process

Each problem entails a variety of symptoms which include but are not limited to:

- Person subject to the difficulty

- Description of the issue
- Timing of the issue

By asking the question 'why' regarding each symptom description, team members can logically develop the related problem chain.

B. Background

Anderson & Fagerhaug (2000) defined root cause analysis as a structural investigation with the purpose of "exploring the real cause of a problem, and necessary actions to avoid it" (p. 10). By using a structured approach causal factors can be discovered based on techniques developed for issue resolution. These techniques can be implemented in two different modes as an input into the decision-making process:

- If root cause analysis is used in a reactive mode, it provides objective identification of organizational process.
- If root cause analysis is used in a proactive mode, it determines and prevents future mistakes.

As Wilson et al. (1993) asserted, "if it could be properly implemented, root cause analysis can show the real reasons for problems" (p. 20).

One of the underpinning features root cause analysis is its inference that bad decisions are based on misled assumptions, erroneous logic, or obsolescent systems. The aim of root cause analysis is to find out why the mistake was committed rather than finding out who did it. Techniqius with a mere emphasis on human error are don't achieve any success in general because of dwindling participation for fear of repercussions. In case of a flawed root cause analysis process, decision errors will continue (Latino & Latino, 1999).

System objectives can include different aspects such as accuracy, timeliness, completeness and relevance which help to define the project scope at a high level. The next step is determining the performance criteria of the new system in the form of clear and measurable goals to be fulfilled by the system.

There is a range of different system constraints. Reality and honesty regarding the resources, scope and schedule go a long way in differentiating a successful system from an unsuccessful one. The triple constraints of scope, schedule and resources depict the interconnectedness as shown in the triangle in Figure 1. While two of these objectives can be achieved easily, achieving a third one can make or break project success.

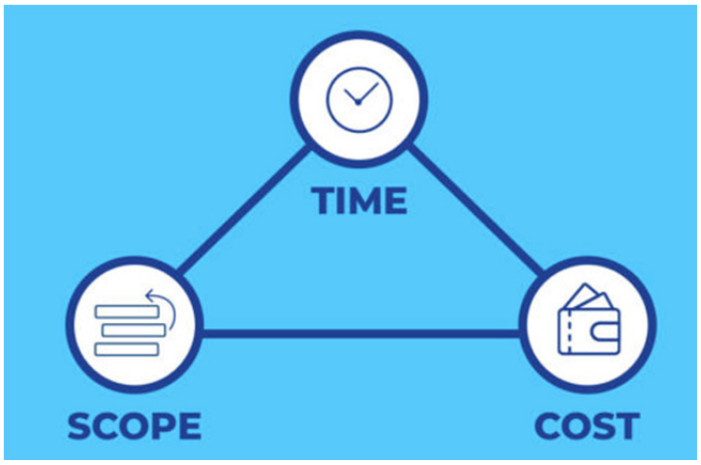

Figure 1. Triple Constraint Triangle

C. Challenges

Design challenges are twofold:

- As a framework of the root-cause analysis can be implemented to heterogeneous devices there must be a platform-independent interface instead of relying on engineering support.
- Change to diagnostic strategies should occur easily by flexibly updating the following layers:
- the core layer,

- the logic layer.

In order to eliminate the re-structuring of the software entirely, it might be useful to put platform-dependent modules within the core layer.

The logic layer can be independently designed and re-used without changing the original application. by merely overwriting HTML files. Still another benefit of classifying the system into two layers is that there is no need to have both layers for some devices.

6. Recommendations

One of the common assumptions for software design problems including design of platforms such as those for troubleshooting is that there always existed a blind spot regarding the process of designing. Yet, without taking all aspects of the design problem into account, such an assumption might be misleading. According to Roozenburg & Cross (1991), the design activities should be considered according to the following dimensions:

- the dynamics of a design process.
- the designer's model
- the design task model
- the design techniques and methodologies combining the model of 'the dynamics of the design process' (see Figure 1).

Throughout the existing body of design research, given the nature of design problems, design issues have mostly been described as being 'wicked' or 'ill-structured' (Cross 1984). Yet, this did not prevent the study of design problems from being relevant. In order to study the structure of design problems in-depth, there should be an emphasis on the following questions:

(1) Which descriptions exist in existing body of literature on design methodology regarding the structure of design problems?

(2) How can a taxonomy of design problems be developed based on explanations?

6.1 Design Problems

Through an initial mapping on existing knowledge of design problems, a more in-depth exploration on 'underdetermined problems' can occur so that these problems can be treated accordingly.

6.1.1 Design Problems as Being Unspecified

Software design activities provides a ground for reasoning based on user requirements. As the underlying reasoning provides no obvious links between the needs and intended artifact there is some degree of underdetermination. According to some scholars, some ways for a design issue to be underdetermined are:

(1) An ongoing needs description (Roozenburg & Eekels, 1995).

(2) Needs and structure are conceptually different categories (Meijers, 2000).

This gap between the 'need' and 'form' would make the solution to a design problem impossible, nevertheless, designers somehow managed to overcome this conceptual gap within their design processes. This is a gradual process, consisting of several steps embedded within the design patterns. Within the light of this information, most design issues can be described as follows:

(1) There are some stabile or unchanging user needs. These 'hard facts' can be discovered upon an initial information gathering and analysis.

(2) Given the vague interpretation of the design issue designers can propose different alternative solutions.

(3) Some part of the design issue arises due to the personal preferences of the designer.

6.1.2 Resolving Software Design Problems According to the Rational Paradigm

There are two main different paradigms regarding the software design methodology. The first one has been developed by Simon (1973) and is referred to as the Rational Paradigm approach. Its features are:

- The rational problem solving process entails a rational search process in which the design problem determines the 'problem space' seeking a design solution:

• A few features regarding the human or Information Processing System do not vary in terms of related task.

• A task environment is conceptualized as a problem space in which problem solving occurs.

• The structure of the task environment specified alternative structures of the problem space.

In case this theory holds true, related problem solving also occurs within a problem space structured by the task environment, which mutually influence related strategies or methods used during the design process.

According to Simon (1973), some potential difficulties might arise in the so called 'ill-structured problems' which have a large problem space so that alternative solutions cannot be listed. Through a rational search process, a 'noticing and evoking mechanism' can be implemented for resolving such wicked issues (Simon, 1973).

6.1.3 Resolving Software Design Problems According to the Reflective Practice Paradigm

Another paradigm was suggested by Schön (1983), who asserted that design should be approached as an activity of reflective practice. According to Schon, a technical rationality might provide some obstacles for related practitioners in the field as they might underestimate the design component of their professions and have a misunderstanding of the nature of human design activities. Based on his 'action-oriented' approach, attention should be provided to the connection among the process and problem structure.

Accordingly, Simon's suggestions regarding the science of design can only be implemented in the case of well-formed problems based on the dynamics of practice. On the other hand, Simon's approach of design as a reflective conversation focuses on the designer's role in terms of specifying the task along with possible solution alternatives by means of 'framing'.

6.2 Kinds of Software Design Problems

In order to understand how design problems are approached by design paradigms is based on their epistemologies.

The decision regarding the application of an 'objective' or 'subjective' interpretation will depend on the designer working on the design issue. According to Dorst (1997), the following factors might influence designer's interpretive behavior:

• Given the requirement to justify the design decisions to stakeholders, related perceptions and ways of interpreting the problem must be made explicit and negotiated among the designer and all related stakeholders in order to avoid subjective interpretations. Given this negotiation process, although ideas and design concepts might still be decided 'subjectively' and implicitly, they are presented explicitly. Therefore, 'objectivity' within the design process can be described an artificial construction by the designer for specific aims.

• In the case of ill-structured problems, 'subjective interpretation' becomes crucial in order to make sense of the problem. Problem structure is developed based on designer's personal preferences and goals. As the major design goal is to create a good design on a reasonable cost and time, the designer can work in both an 'objective' and a 'subjective' mode.

• In order to establish a shared understanding within a group design process to keep track of everyone, it might be difficult to achieve a consensus amidst various objectifying statements and arguments (Valkenburg 2000).

• As a specific design project provides the freedom of choice to the designer to some extent, design becomes a subjective activity, which can be best explained in terms of reflective practice. While this subjective interpretation mostly holds true for the conceptual phase of many design processes, it could also extend over the whole design process.

6.3 The Issue with Design Problems

While reflecting upon the types of underdetermination regarding the design process itself may not help us in terms of developing a structure for problems, it makes the development of a taxonomy of design problems more difficult due to the dual nature of design. A structure could be established by finding related pattern of connections among sub-problems. According to the Reflective Practice paradigm, the problem structure depends on alternative solutions, and cannot be objectively developed.

By looking at the designers' sub-problems they encounter in their every day lives, a method for the empirical study of design could be developed as contextual problem solving.

6.3.1 Co-Evolution

According to Dorst & Cross (2001), rather than trying to fix the issue based on a framework and looking for a reasonable solution concept, creative design entails constant iteration of analysis, synthesis and evaluation processes between the problem space and solution space. In other words, the designer tries to match a problem-solution pair based on a 'co-evolution' of the problem and the solution. Even though the design process cannot be approached as something automatically leading from a problem into a solution this should not deter us from theorizing about design at all.

6.3.2 Design Problems as Contextualized Problems

Both paradigms emphasized the ability of computer systems to find a solution to ill-structured problems by developing formal procedures that manipulate relevant representations of the real world to solve a problem. Despite being based on a rational approach, both of these dominant paradigms proved to be useful to a limited extent.

According to Dreyfus (2003), problematic situations that occur as a result of a ‘breakdown’ during the fluent problem solving behaviour provide ample opportunities for real choice as these breakdowns can become turning points depending on the designer’s ‘objective’ or ‘subjective’ interpretation of the situation at hand. Depending on his reflective conversation with the situation at hand, the designer can uncover implicit or unknown structure of the design issue by means of related structuring actions.

6.3.3 Levels of Expertise

Dreyfus (2003) claimed that the nature underpinning a problem for which a solution needs to be found depends on the problem solver’s level of expertise. These can be summarized as follows:

(1) A novice will need to follow a set of fixed rules to cope with the issue at hand by taking into account situational factors objectively.

(2) As the novice makes transition into an advanced beginner, he realizes that the situational aspects play a crucial role so that maxims are used as a guidance during the problem situation.

(3) A competent problem solver tries to accomplish its goals based on a choice of plan depending on the relevant aspects of a situation chosen. There can be a trial-and-error process accompanied by a feeling of responsibility. In addition to this, learning and reflection upon learning is also evident at this stage.

(4) Once a problem-solver becomes proficient he is able to identify the most critical issues and develop an appropriate plan accordingly.

(5) The real expert displays the appropriate action straightaway based on his intuition so that problem solving and reasoning are not even apparent at this level of working.

All of these different ways of looking at the problem situation can co-exist together as there would rarely be someone who would have an expertise in all of the design aspects While novice designers might prefer to follow strict guidelines according to some guidance, more advanced ones might prefer to get their interpretation and reflection involved in the design process.

Based on the level of expertise, the nature of the design problem might be approached differently which also affects the paradigm choice for supporting design processes.

6.4 Empirical Study

Needless to say, such a research method should be neutral in the sense that it only provides a ‘language’ and a method of working to keep track of the process of design as closely as possible. By being independent, the method provides an opportunity to delve into the details of the design behaviour to recognize the related patterns of the problem without being completely problem-specific (case-specific). The patterns of links as perceived by designers could be traced with ‘linkography’, as suggested by Goldschmidt and van der Lugt (2001). For this purpose, a ‘link matrix’ can be developed. The number of matrixes used does not necessarily need to be restricted to one as there might be cases in which one matrix would not suffice. To give a specific example;

- One matrix in which a subset of design issues are located on two axes
- Another one where the solution elements are on both axes
- Another one in which the subset of issue located on one axis relate to the solution elements on the other

Filling such matrixes would guide the researcher in keeping track of the perceived problem and the design solution so that patterns in the designer’s operations can be distinguished from design problems and solutions.

7. Conclusion

This research investigated the use of a trust-based troubleshooting platform which utilizes the collaboration of chatbots and end-user devices by utilizing the technique of root-cause analysis. One of the main advantages has been the use of multiple communication interfaces by recent devices. In the case of an interface use, a probe request can be conveyed to other devices by means of another interface. Finally, applying a design based approach rather than a root cause analysis might prove more beneficial for software engineers in order to establish user trust. More research in adopting the same platform for other IoT technologies and ways to overcome potential implementation challenges would be useful. In order to explore the methods for explaining the structure of software design issues, a priority taxonomy of software design problems should be left aside as doing so would emphasize the structures of reasonably specified problems. As software design problems are largely underdetermined, such a method would be futile. In order to grasp the details, problems should be approached as being situated as if they were seen through the software designer's eyes. In order to understand the way a software designer tackles a problematic situation a model of software design expertise can be utilized which should be validated by further empirical research.

References

Aggarwal, B., Bhagwan, R., De Carli, L., Padmanabhan, V., & Puttaswamy, K. (2011, December). Deja vu: fingerprinting network problems. In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies* (p. 28). ACM. https://doi.org/10.1145/2079296.2079324

Boyaci, O., Beltran, V., & Schulzrinne, H. (2010, December). Bridging communications and the physical world: Sense everything, control everything. In *2010 IEEE Globecom Workshops* (pp. 1735-1740). IEEE. https://doi.org/10.1109/GLOCOMW.2010.5700238

Cui, H., & Biersack, E. (2011). Trouble shooting interactive web sessions in a home environment. *Proc. of HomeNets '11*, Toronto, Ontario, Canada, Aug. 2011. https://doi.org/10.1145/2018567.2018574

Dong, C., & Dulay, N. (2011, August). Argumentation-based fault diagnosis for home networks. In *Proceedings of the 2nd ACM SIGCOMM workshop on Home networks* (pp. 37-42). ACM. https://doi.org/10.1145/2018567.2018576

Kim, K. H., Nam, H., & Schulzrinne, H. (2014, April). WiSlow: A Wi-Fi network performance troubleshooting tool for end users. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications* (pp. 862-870). IEEE. https://doi.org/10.1109/INFOCOM.2014.6848014

Kim, K. H., Singh, V., & Schulzrinne, H. (2011). DYSWIS: Collaborative Network Fault Diagnosis-Of End-users, By End-users, For End-users. *Columbia University Technical Report cucs-017-11*, 2011.

Rayanchu, S., Patro, A., & Banerjee, S. (2011, November). Airshark: detecting non-WiFi RF devices using commodity WiFi hardware. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference* (pp. 137-154). ACM. https://doi.org/10.1145/2068816.2068830

Sundaresan, S., Grunenberger, Y., Feamster, N., Papagiannaki, D., Levin, D., & Teixeira, R. (2013). WTF Locating Performance Problems in Home Networks. *SCS Technical Report GT-CS-13-03, 2013*

Wustner, S., Joumblatt, D., Teixeira, R., & Chandrashekar, J. (2012, December). Automated home network troubleshooting with device collaboration. In *Proceedings of the 2012 ACM conference on CoNEXT student workshop* (pp. 61-62). ACM. https://doi.org/10.1145/2413247.2413284

Appendix A- Sample Root-Cause Analysis

Title

Grade

Issue Summary

Priority

Severity

Root Cause

Corrective Action

Preventive Measures

<u>Call log available for only last half a year. All older calls lost.</u>

Needs improvement

A performance and storage issue not controlled by Dialer. A framework limitation of 500

P4

S2

People: Engineer misclassified the issue as a feature request for increasing framework limitation rather than performance issue.

Improving communication between team members would remove ambiguity in such issues.

People Improvement: Avoiding miscommunication with client/internal team

<u>Group MMS message appearing in 1:1 conversation. Same for other group members using Messages on Wi-Fi.</u>

Good

The default group messaging settings is "To Send an MMS reply to all recipients (Group MMS)", duplicate issue

P2

S2

People: Engineer updated the original bug as duplicate which was referring to an intended behavior rather than an issue to be fixed.

The team should be clear before asking for update on any bug to ensure if fix is already communicated and if issue is still not fixed, then the original bug should be reopened.

People Improvement: Better communication with client/internal team

<u>Call Drop: Error Unspecified</u>

Needs improvement

Due to setup unavailable, the issue was assigned to dialer-test-triage, this was sent to Telecom for having another look.

P4

S4

Tools:

- Non availability of enough testing devices and no access to production for simulating and troubleshooting the issue

Process:

- Lack of required information while triaging the bug

A process should be in place regarding cases when issues cannot be reproduced due to the lack of setup.

Process improvement for cautiousness during ticket updates

Tool improvement for making setup available for testing

<u>DUT will not display "delivered" when NFS at the first time (SMS delivery report has been enabled.)</u>

Needs improvement

In case of NFS, RCS client expects the server sends IMDN Delivery Notification if SMS is delivered. However, since the session is closed, no IMDN message is received. Google Messages relies on the carrier network to send back IMDN Delivery Notification after the SMS is delivered. In this case, such notifications were not received. As Vodafone does not support NFS this is a network issue rather than a client issue. Suggestion is to test with a carrier network that support NFS.

P1

S2

<u>People:</u> Team didn't have detailed information on the query asked several times by the client regarding a potential design limitation and could provide a response only one month later which turns out to be a network issue.

In order to provide a more appropriate analysis, engineers should be equipped with a full understanding of the differences between design limitations and network issues.

People Improvement: Trainings/ Knowledge gaps

<u>Missed calls don't show up until reboot, phone gets in a wonky state when this happens</u>

Good

No notifications for missed spam calls and not applying any special treatment to spam calls which are not blocked.

P4

S2

<u>Process:</u> When a new ticket is raised logs are asked to check whether issue can be reproduced or not, yet whether it is a duplicate ticket is only being checked at the very end which may render initial works redundant.

A process should be in place regarding cases whether team will raise new ticket if there are already duplicate tickets and at what stage they should check whether the ticket is duplicate or not.

Process improvement for cautiousness during ticket updates

<u>-There is no share contact option in message APP</u>

Fail

Instead of receiving the images/video/audio and contact that were sent from DUT, there is no share contact option in message APP. Share contact feature is added in v3.9. It is suggested to test it with that version once it's available.

P1

S2

<u>People:</u> The status was marked as obsolote and the client was provided with a short response of testing the feature in v3.9 which would not to be very satisfactory given the detailed bug report sent by him.

The team should have updated the client with more specific response as the first priority is to make client comfortable and understandable .

People Improvement: Avoiding miscommunication with client/internal team

<u>[User Feedback] Conversation is not loading and unresponsive (clicking on the top-right three dot menu button</u>

does nothing.

Good

In Tachyon backend,there are traces about the request Tachyon send to Bugle and response (ack) from Bugle. But the traces about the request Tachyon send to Ditto and response (ack) from Ditto cannot be checked since they are using a different trace_id. More logging for this trace_id (Tachyon->Ditto and Ditto ack to Tachyon) will be added.

P2

S2

Tool: Lack of the trace of Tachyon->Ditto and Ditto back to Tachyon results in the bug being marked as infeasible.

A process should be in place when to mark the status as infeasible despite the detailed information provided by the client and how to communicate this decision to the client.

Process improvement for cautiousness during ticket updates

Tool improvement for trace_id logs (Tachyon->Ditto and Ditto ack to Tachyon).

Appendix B- General Troubleshooting Best Practices

Process

Step 1: Ticket sanity check

Follow the Dos and Don'ts

Step 2: Reproduce

Clearly mention the 'Steps to Reproduce' in the issue/ticket

Step 3: Logs Analysis

Properly do the Initial log analysis of the issue

Step 4: Blocking

Mark the blocker release issues and blocking the user experiences. Add it to the blocking hotlist for the release.

Step 5: Set Priority

Prioritize the issue based on various factors that are mentioned in pre-triage guide

Step 6: Assign to Pillar

Assign the issue to the respective team for further investigation

Note: Mention Blocked By/Duplicate to the appropriate issues which saves time and allows better management of Bugs.

Step 7: Issue tracker component

Raise internal ticket in the relevant component

Step 8: Special cases

For localization issues, follow pre-triage guide

Step 9: Final Check

Walk through and check all the aspects required to close/conclude the issue.

Do's and Don'ts

Do's:

- Make sure to fill in the '**Found in**' custom field while triaging a ticket for both internal and external tickets
- While selecting the custom field for internal ticket:
- Value: "**More info needed"** - state the exact information that is required to triage the ticket which is understandable for everyone.
- Value: "**Must Fix**" - High priority fix
- Value: "**Fix in current release**" - The ticket that needs to be fixed on high priority in the current release
- Value: "**Fix in future release**" - The ticket that needs to be fixed with medium priority.
- Value: "**On Product roadmap**" - The ticket that needs to be fixed in the next 6 months. PS team can close the external ticket.
- Value: "**Low priority - Cannot fix**" - The external ticket can be closed by t1 team with appropriate comments.
- Follow up with the **Eng team** about the targeted release for fix and fill in the '**Targeted to**' custom field for both internal and external tickets.
- Make sure to give appropriate comment while closing a bug. If it is being closed because it is a low priority issue and we do not intend to fix it, also mark the '**Low priority**' custom field before closing.
- If an external bug is a duplicate of an existing bug, the bug should still be kept open and partner should be provided with updates.
- Assign the ticket to appropriate hotlists e.g. **Massage need info** as required.
- While raising the internal ticket, always use the appropriate base component.
- If the bug is related to Android OS behavior, does the bug indicate if Android Messages was the default SMS app?
- If not, ask the reporter to check their default app.
- If the bug relates to sending/receiving messages, is it clear if they were SMS, MMS, or RCS (Rich Content Services)?
- If not, ask the reporter to clarify. An easy check is if the send button on that conversation says "**chat**" then it's RCS, otherwise it's *MS.
- If the bug relates to RCS, is the environment indicated?
- If not and the reporter is an RCS team member, ask them to clarify.

Reporters outside the RCS team do not need to indicate environment if it isn't already in the big.

- If the bug relates to RCS, does it indicate if the user was on WiFi or mobile data?
- If not, ask the reporter to clarify.

Don'ts:

- Never mark an external ticket as a dupe to other external or internal ticket
- Never mark an external ticket as blocking to an internal ticket or other external ticket raised by a different partner.

General Issues

Scenario #1A: Group Message Issues- No Display of Edit Status

When you open the display input status switch in a single chat or group chat session and enter the message, the other party does not display the edit status.

Issue Analysis: The typing indicators are not present on some devices. It seems like platform UI is ignoring intents sent by RCS engine.

Resolution: "**BugleAction:** The GLOBAL Rcs handler logs all events that it ignores so we don't accidentally miss any. The typing indicators are to be received by the contact picker event handler, and the conversation event handler. We should special case this event (and capabilities) in the global RCS handler so that people stop mentioning this log line as a cause for missing typing indicators and capabilities requests. There's no reason to believe this log line is related to the observed effects in this bug.

Scenario #1B: Group Message Issues- Delay in Receiving Group Chat

Receiving group chat is delayed. The user got disconnected while in a group chat, from the below logs we can observe that session got disconnected.

Issue Analysis: This issue seems to be a race condition.

- User A sends an initial message which was delivered to User B and also sent another message at the same time. When B came back online, B couldn't find stored messages.
- Since there were no stored message, server didn't trigger a rejoin to the conference.
- Hence, all subsequent messages will be delivered only after User B rejoins.
- The rejoin for User B happened only after User B sent a new message (which triggered a new INVITE)
- At that point all the stored messages for User B also got delivered.

Resolution: Session got terminated might be the reason for delay in chat messages.

Please check the timeStamps.

Scenario #2A: Provisioning- Registering RCS according to GSMA

When a message is a non-default SMS application, you may hope that RCS can be registered according to the GSMA protocol. Otherwise, downloading the RCS client on the some RCS inventory phone will result in a double-stack collision problem.

Issue Analysis: You should request for the bug report (Verbose enabled), issue screenshots, and reproducibility rate X times out of Y times.

Verbose Logs:

- adb shell setprop log.tag.Bugle VERBOSE
- adb shell setprop log.tag.CarrierServices VERBOSE
- adb shell setprop log.tag.BugleRcsEngine VERBOSE

Please run the above command lines in ADB shell and then collect the bug report after executing the above commands.

If Else: Ask customer to update the AM and CS application to the most recent version which will be available in Play Store.

Resolution: You should ask the customer to make AM as their default messaging App.

Scenario #2B: Provisioning- From Scratch or SIM Swap Scenario

Provisioning not working when forcing it again from scratch or SIM Swap scenario.

Issue Analysis: You should request for the bug report (Verbose enabled), issue screenshots, and reproducibility rate X times out of Y times.

Verbose Logs:

- adb shell setprop log.tag.Bugle VERBOSE
- adb shell setprop log.tag.CarrierServices VERBOSE
- adb shell setprop log.tag.BugleRcsEngine VERBOSE

Please run the above command lines in ADB shell and then collect the bug report after executing the above commands.

If Else: You should ask the customer to update the messaging application to the most recent version which will be available in Play Store.

Resolution: You should ask the customer to make AM as their default messaging App.

Scenario #3A: Notification Issues- On/Off Option Not Supported

Notification on/off option is not supported on Messaging 3.0.040.

Issue Analysis: The issue is with an older version of Messaging. The user was not able to understand the navigations in the: Settings menu" for notifications.

Resolution: You should request the user to install the latest version of Messaging Nougat.

You should also suggest the required setting navigations to the user for some other OS devices.

Scenario #3B: Notification Issues- Request for SIM2 Sound Notifications

SIM2 notification sound item is none while the user is requesting for two different sound notifications for the two SIMS mounted on to the device.

Issue Analysis: The issue is not with the messaging app, it is device specific to set two different sound notifications to the two different SIMs, so it is intended behavior from Messages.

Resolution: You should observe the internal teams suggestion on the bug, if it is going to be fixed or guided with any further suggestions on the settings.

Scenario #4A: Server Connectivity Issues- IMAS Connections Dropped

IMAS connections got dropped.

Issue Analysis: There is a drop in IMAS active connections, same is verified in tool. When you check for the "IMAS server active connections" you will see that there is a downfall in the activity. So, you should raise an internal ticket to the SRE team and an external ticket for external customer communication.

Resolution: Once the internal team marks it as fixed, then the T1 team should verify if the issue is fixed or not via tools and confirm the same to the customers.

Scenario #4B: Server Connectivity Issues- Lost RCS Connection

RCS connection is lost after updating/install same build.

Issue Analysis: Logs show that the service was about to shut down and unregistered TelephonyChangeReceiver. Yet, then it immediately restarted using the same instance of SimStateTracker so that it missed SIM LOADED event which should have triggered provisioning and then registration.

Resolution: Unregister SimStateTracker together with TelephonyChangeReceiver when JibeService is shutting down. So that next time when JibeService is starting we make sure that TelephonyChangeReceiver is registered before SimStateTracker and never miss SIM LOADED event.

Made sure SimStateTracker is not unregistered for JibeService running in CS.apk. Unfortunately, there is no chance to add more good tests RcsService because it will require moving of RcsServiceTest into a correct package (wrong now) and refactor all the tests to avoid Context mocking.

Scenario #5A: SMS/MMS Fallback Issues- Duplication

Both SMS and RCS message are received (duplicate).

Issue Analysis: As the receiver has activated the DOZE mode in the device the message sent has got delayed in delivery. The client is currently not sending delivery reports for incoming messages when in doze mode.

Resolution: AM has to send delivery reports for incoming messages even when the DOZE mode is enabled in the device.

Scenario #5B: SMS/MMS Fallback Issues- Switching to Chat

Messaging client is switching to chat for Non RCS contact.

Issue Analysis: The issue is triggered when the client receives an error response to a user who was previously as known RCS users. From that point in time onward, all OPTIONS error responses will be interpreted as "RCS capable user", even if the user is not.

Resolution: You need to check connectivity issues, doze mode and check whether the end user is RCS enabled or not, check in server logs - before checking the client logs. Client recovers upon process restart.

Copyrights