COCOA 不具合問題

COCOA bug problem

増田大輝
名古屋工業大学大学院工学研究科
工学専攻博士前期課程　情報工学系プログラム

Hiroki MASUDA
Nagoya Institute of Technology
Department of Computer Science

【Key words】
1. アプリケーション(Application)
2. 厚生労働省(Ministry of Health, Labour and Welfare)
3. 多重下請け構造(multiple subcontract structure)
4. 技術者倫理(Engineers' ethics)
5. 公衆衛生(Public health)

1. 諸言

スマートフォンは2010年より急速に普及し，日本では2019年度に世帯保有率が8割を超える[1]など，我々にとって最も身近な通信機器の一つである．一般的に，スマートフォンユーザは各々必要なアプリケーションを適宜インストールして使用する．そのため，スマートフォンの普及に伴い，アプリケーションのダウンロード数も年々増加している[2]．

しかし，スマートフォンの登場によりIT市場が急拡大した反面で，IT技術者の量と質に不足が生じていると考えられている[3]．特に，技術の移り変わ

りが早いスマートフォン向けのアプリケーション開発においては，こうした問題はより顕著であると推測される．そして，アプリケーション開発人材の不足は重大なバグや不具合を発生させる根本的な原因となり得る．

本稿では，マネジメント側の人員不足と開発者側の当事者意識の欠如が招いた問題として，2020 年 9 月から翌年 2 月にかけて発生した，新型コロナウイルス接触確認アプリケーション「COCOA」の不具合について紹介する．この不具合について，その直接的原因と経過を分析し，さらにその背景にあった開発体制の問題点を指摘する．これらに対して倫理的問題点や改善策について議論する．

2.　COCOA と不具合の概要

2-1　COCOA について

新型コロナウイルス(COVID-19)は 2019 年 12 月以降に中華人民共和国湖北省武漢市を中心として世界中に感染が拡大したウイルスである．このウイルスの大きな特徴の一つとして，感染力が非常に高いことが知られている．日本においても，2020 年 1 月以降継続的に感染者が発生し，感染者数は増加と減少を繰り返しながら推移している．

新型コロナウイルスの感染拡大を防止するために，Apple と Google は 2020 年 5 月より Bluetooth ベースで濃厚接触の検出と追跡を可能とする API (EN API: Exposure Notification API) の提供を協同で開始した[4][5][6]．この API を利用できるのは，各国の公衆衛生当局が配信する一つのアプリケーションのみという「一国一アプリ」の原則が提示され，日本では厚生労働省から COCOA が配信されることとなった．

2-2　不具合の概要

2020 年 9 月 28 日の COCOA バージョン 1.1.4（以下，v1.1.4 と表記）の配布後，Android 端末において陽性者と接触したユーザに一切の通知が表示されない不具合が発生した．一方，iOS 端末では正常に通知を受け取れていた．本不具合は同年 11 月にインターネット上のソフトウェア開発プラットフォ

ーム「GitHub」上で指摘があったものの，翌年 1 月ごろに SNS や各社報道で多数の報告がなされるまでは十分に認知されなかった．本不具合は 2021 年 2 月 18 日のバージョン 1.2.2 により修正されるまでの約 4 ヶ月間に渡って放置され続けた[7]．

3.　COCOA の仕組みと不具合の詳細[7]

本章では，COCOA の仕組みと不具合の詳細について説明する．

3-1　COCOA の仕組みと濃厚接触の判定・通知の処理について

本節では，COCOA の大まかな処理の流れについて述べた後に，本不具合に深く関連する濃厚接触の判定・通知の処理について詳しく説明する．

COCOA では次の手順で処理を行う．1）まず，COCOA がインストールされた端末同士の接触情報を端末に保存する．2）陽性者が発生した場合には，その情報が通知サーバに登録され，各ユーザは通知サーバよりこれらの陽性者情報を取得する．3）取得した陽性者情報を各端末に保存されている接触端末情報と突合する．4）突合の結果によって濃厚接触の判定並びに通知を行う．

本不具合には上述の 2 以降の処理が大きく関わっている．この部分の処理について，より詳細に説明する．a）まず，陽性者情報として診断鍵と接触リスク計算用パラメータを通知サーバより取得する．b）a で取得した情報を EN API に渡す．EN API では接触リスク等を計算し，サマリー情報（接触の有無・最大リスク）を計算する．c）EN API から返却されたサマリー情報より各端末内で濃厚接触の判定と通知を行う．d）濃厚接触に係る詳細情報をまとめてアプリ内で表示する．

3-2　v1.1.3 以前の不具合

本稿で主に取り扱うのは v1.1.4 以降の不具合だが，これを説明するために本節では v1.1.3 以前に存在した不具合について説明する．COCOA の v1.1.3 以前には主に 2 点の不具合が存在していた．まず，陽性者と接触した旨のプッシュ通知が来るのに詳細を確認できない不具合が Android 版と iOS 版の双方に存在した．これは c における濃厚接触判定に問題があり，「単なる接触」を

全て通知していたことに起因する．次に，Android 版では全ての接触者情報を表示してしまう不具合が存在していた．これは d における詳細情報取得に問題があり，接触者全ての情報を取得して表示していたことが原因となっていた．これらの不具合を解決するために，次節で取り上げる v1.1.4 がリリースされた．

3-3　v1.1.4 以降の不具合

前節の v1.1.3 の不具合を修正するために v1.1.4 では c の濃厚接触判定に設定が追加された．この判定処理では接触リスク計算用パラメータを元に算出された最大リスク値が設定された閾値を超えるか否かで濃厚接触を判定する．この改修の結果，iOS 版では動作が正常になった一方で，Android 版では通知も表示も一切出力されない不具合が発生した．次節では，本不具合の直接的原因について説明し，なぜ Android 版でのみ一切プッシュ通知やアプリ内表示が出力されなくなったのかを明らかにする．

4.　不具合の原因[7]

本章では，不具合の原因について説明する．

4-1　不具合の直接的原因

本不具合の直接的原因はリスク値(Total Risk Score)計算におけるバグにある．このリスク値は四種類のリスクパラメータと呼ばれる値から算出される．これらはそれぞれ①感染リスクパラメータ，②接触期間パラメータ，③経過日数リスクパラメータ，④減衰リスクパラメータと呼ばれる．これらのパラメータを用いると，Total Risk Score は以下のように計算される(計算式は参考文献[7]８頁の記述を参考に作成)．

$$\text{Total Risk Score} = r1(①) \times r2(②) \times r3(③) \times r4(④).$$

ただし，$r1, r2, r3, r4$は各パラメータよりリスク値を算出する関数とする．この Total Risk Score が 21 を超えた場合に，COCOA では濃厚接触の通知とアプリ内表示がなされるように設計されていた．

リスクパラメータ①はもともと日本では用いない予定だったものの，v1.1.4 の改修に伴いリスク値の計算に使用されることになった．そのため，デフォルト値の 0 がそのまま使用されており，設計の段階では 0 を入力すると関数 $r1$ からは常に 7 が出力されることを意図していた．しかし，Android ではリスクパラメータ①が 0 の場合，関数 $r1$ の返却する値は常に 1 となる仕様で，この場合，他の 3 つのパラメータの値に関係なく Total Risk Score は常に 21 以下となる．したがって，Android 端末では濃厚接触の判定が一切行われず，それによってプッシュ通知やアプリ内表示が出力されなくなったのである．

一方で，iOS では，意図した通りに関数 $r1$ から 7 が返却されていたので，正常な動作を示したのである．後に COCOA 不具合調査・再発防止策検討チームの報告書では，厚生労働省の CIO 補佐官が「Google と Apple の EN API の仕様の差異に関して，両者それぞれの回答として差異はないという話を聞いていたので，その回答ベースで判断してしまった」([7] 23 頁)とも述べており，仕様が完全に共通していると信じ込んでいた．

4-2 テストについて

通常，新しいバージョンがリリースされる前にテストを実施することで，バグや不具合が明らかになり，修正される．COCOA においては，テスト環境が整備される前は，接触通知までの一連の結合テストを実施できなかった．そのため，接触通知を含めたテストは 2020 年 10 月 12 日にテスト環境が整備されてから実施され，v1.1.4 がリリースされる前に接触通知を含めた結合テストが行われることはなかった．しかし，テスト環境整備後の結合テストでは，テストを効率的に実施する観点から，接触通知が発生しやすい状態で実施された．具体的には，Total Risk Score が閾値 1 を超えれば接触判定と通知がなされるように変更されてテストが実施された．これにより，本来は問題があったはずの Android 版において不具合が顕在化しなかったのである．こうした経緯で，テスト時に本不具合を発見できず，長期間に渡って Android のみで一切通知と表示が出ない状況を開発者側が認知できない状態に陥った．

5. 開発体制と倫理的問題

5-1 GitHub 上での指摘の長期間放置[7]

2-2 節でも述べたが，本不具合は世間一般に周知される以前に，2020 年 11 月 25 日時点でインターネット上のソフトウェア開発プラットフォーム「GitHub」上で指摘されていた．GitHub では，Issue と呼ばれる機能が存在し，プログラム中に含まれる不具合やバグなどについて指摘できる．その後，同年 12 月 4 日に開発を担当した下請け会社が，当該指摘を改善の検討リストに追加していた．しかし，当該指摘は SNS や報道等で取り上げられるまで放置され続けた．このような状態を生み出した原因として，業務フローが曖昧なまま進められていた杜撰な開発体制が指摘されている．

厚生労働省によると，GitHub にプログラムを公開したのは，アプリケーションの透明性を担保するためであり，もともとは GitHub 上でなされた指摘などについては取り合わない方針だった．そのため，委託事業者や再委託事業者との契約もその方針に基づいて，GitHub 上での指摘に取り合うことまでは含まれていなかった．しかし，厚生労働省は一転して，委託事業者に対して元々の契約には含まれていない Issue の管理をする依頼を行った．これを受けて，委託事業者は実際に開発を担う再委託事業者に対して重要度を確認しておく旨の連絡を行った．しかし，再委託事業者は Issue から転記して検討リストを作成するまでが自らの業務範囲であると認識していたため，重要度の確認及び優先順位の決定は別の業者が行うものだと考えていたことが後の調査で明らかになっている．以上より，本来は存在していなかった「GitHub の Issue の管理」というタスクが新たに追加された際に，誰がいつどのようにやるかが明確にされていなかったのが問題点である．

5-2 開発体制の問題点

COCOA 不具合調査・再発防止策検討チームの報告書では，前節で述べたように業務フローが曖昧になった原因についても複数言及されている．まず，プロジェクトマネジメントによる明確な役割分担が行われていなかった点が挙げられている．報告書では，「厚生労働省は委託事業者が品質管理・保証」([7] 32 頁)など開発物に係る責任を負っているものだと認識していたのに対して，「委託事業者はプロジェクト管理の一部や工数管理の一部」([7] 32 頁)を

担当していると認識していたと述べられている．このことから，マネジメントの権限が厚生労働省か委託事業者のいずれにあるのかが不明瞭だったと考えられる．次に，事業者間のコミュニケーションが不足していたことが指摘されている．報告書では，「各々が『他がやっているだろう』という思い込みを持っていた」([7]30 頁)状況にあったと述べられている．事業者間の役割分担や担当業務が不明瞭だったために，実際の環境でのテストの検討・提案が行われなかったことも不具合を未然に防げなかった原因でもある．このように，業務のマネジメントを誰が担うかが厚生労働省と事業者間で不明瞭だった上に，事業者間でも責任の所在が不明なまま開発が進められたのが問題と考えられる．

5-3 倫理的問題

本節では，厚生労働省と委託事業者並びに再委託事業者について，倫理的問題とそれを引き起こす要因となっている現代日本におけるソフトウェア開発の問題点を指摘する．

前節で述べたように，国民の健康と公衆衛生の維持に責任を担っているはずの厚生労働省がCOCOAの開発を委託事業者に丸投げしていたという点は倫理的問題と言える．これには，厚生労働省内にアプリケーション開発の専門家が不足していたのは勿論のこと，そもそも同省が慢性的な人員不足の状況にあったことが問題点として挙げられる[8]．

また，委託事業者が業務の大部分を複数の事業者に再委託し，厚生労働省がこれを認可した点も問題である．このように，業務が再委託や再々委託される構造は多重下請け構造と呼ばれている．多重下請け構造の問題として，一つのプロジェクトに多数の事業者が関係することで，責任の所在が不明瞭になることが挙げられる．厚生労働省では，契約金額の 50%以上での再委託を原則禁止することで，多重下請け構造を防止している．しかし，COCOA の開発プロジェクトに関しては，契約金額の 94%での再委託を許可していた[9]．

最後に，開発を担当していたエンジニアの当事者意識が欠如していた点を指摘する．日常的に GitHub を用いて開発を行なっていると，Issue に指摘が追加された場合には，開発に携わっているエンジニアに通知が送信される．そのため，本不具合が長期間放置されたことは，エンジニアが確認を怠った

ことを意味すると考えられる．指摘の確認は自分には関係がないという認識がエンジニア達に蔓延していたのではないかと推測される．

厚生労働省が事業者任せだったこと，多重下請け構造を許したこと，エンジニアの当事者意識が欠如していたことが原因で，誰も責任を持たず，誰も不具合に気付かない状態に陥っていたと考えられる．

5-4 改善策

本節では，前節で挙げた 3 つの問題点について改善策を提案する．

まず，厚生労働省が事業者任せだった点については，同省に専門家がいなかったことが問題であると考えられる．そのため，政府側でアプリケーション開発の専門家を用意してプロジェクト全体から詳細までを把握するのが効果的である．例えば，外部の有識者や内閣参謀の IT 総合戦略室，2021 年 9 月に創設が予定されているデジタル庁から人員を確保するなどの改善策が考えられる．

次に，多重下請け構造の解消により，開発責任と役割を明確にする必要がある．これを実現するには，再委託を可能な限り減らし，一社で一貫して開発を行うのが望ましい．

そして，当事者意識と責任感を持ったエンジニアを増やすためにも，技術者への倫理教育を充実させる必要がある．具体的には，大学などの教育機関や企業内教育を拡充させることで，責任感のある技術者を育成する．COCOA の開発プロジェクトでは，自分の作ったプロダクトが国民の健康を守っているという意識をエンジニアが持ち，不具合を見逃さず，指摘に素早く対応するのが理想的である．

6. 結言

本稿では，2020 年 9 月 28 日以降 4 ヶ月間に渡って放置された COCOA 不具合問題について述べてきた．本件は，陽性者と接触した Android 端末に通知が送られない不具合で，その直接的原因は，仕様の確認不足により発生した数値処理のバグである．また，実際よりも条件を緩めた状態での不十分な

テストを実施したことで，不具合を見抜けなかった．さらに，指摘あったにも関わらず，素早い改修が行われなかった．倫理的問題としては，厚生労働省の事業者任せの姿勢，多重下請け構造を許容したこと，エンジニアの当事者意識の欠如が挙げられる．複数の組織それぞれに原因があり，本件はIT業界に蔓延する問題が顕在化した一例とも言える．このような問題を再び起こさないようにするには，情報工学の知識を有する人材をマネジメント側にも増やし，開発側では技術力と倫理観を兼ねそろえたエンジニアを増やすという中長期的な取り組みが必要となる．技術者を目指す我々は，本件を教訓に責任をもってものづくりに取り組むことが求められている．

[参考文献]
[1] 総務省, “令和２年版　情報通信白書　情報通信機器の保有状況”, https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/r02/html/nd252110.html
[2] 総務省, “平成２８年度版　情報通信白書　モバイル向けアプリ市場”, https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h28/html/nc122230.html
[3] 独立行政法人情報処理推進機構　社会基盤センター, “「IT 人材白書 2020」概要”, https://www.ipa.go.jp/files/000085256.pdf .
[4] Apple, “Apple と Google、新型コロナウイルス対策として、濃厚接触の可能性を検出する技術で協力”, https://www.apple.com/jp/newsroom/2020/04/apple-and-google-partner-on-covid-19-contact-tracing-technology/ .
[5] Google, “濃厚接触の可能性を通知するシステム: 新型コロナウイルス感染症（COVID-19）の拡大抑止に取り組む公衆衛生機関をテクノロジーで支援する”, https://www.google.com/covid19/exposurenotifications/ .
[6] IT メディアニュース, “Apple と Google、新型コロナ「暴露通知」の API 公開　日本を含む 22 カ国がアクセス済み”, https://www.itmedia.co.jp/news/articles/2005/21/news048.html .
[7] 厚生労働省, “接触確認アプリ「COCOA」の不具合の発生経緯の調査と再発防止の検討について”, https://www.mhlw.go.jp/content/000769774.pdf .
[8] IT メディアニュース, “平井大臣、COCOA に苦言「出来のいいアプリではなかった」「発注にも問題あった」”, https://www.itmedia.co.jp/news/articles/2102/10/news124.html .
[9] 中日新聞（令和３年２月２日）, “COCOA 開発 94%再委託　規定超過、不具合の原因把握困難”, https://www.chunichi.co.jp/article/205422 .

＊リンクの最終確認は 2021 年 7 月 31 日