接触確認アプリ COCOA からの教訓

応
般

楠　正憲 | 内閣官房

2021 年 1 月 Android 版の接触確認アプリ COCOA が数カ月にわたって動作していなかったことが明らかにされた．筆者は 2020 年 4 月から接触確認アプリの導入について，有志での議論に参加し，有識者会議のメンバとして，また途中から政府 CIO 補佐官として，接触確認アプリの導入を支援してきた．本稿では接触確認アプリ COCOA の開発と運用について，どのような課題があったかについて振り返る．

接触確認アプリ導入の経緯

筆者が接触確認アプリについて知ったのは昨年 (2020 年) 3 月頃のことである．ちょうどシンガポールの Trace Together が話題となって，日本でも接触確認アプリをリリースできないかといった話題で，いくつかのコミュニティが盛り上がり始めた．Android のシェアが高いシンガポールに対して，日本では iPhone のシェアが非常に高く，iPhone が Bluetooth のバックグラウンド通信を認めていないことから，実際に日本で接触確認アプリを機能させようとすると iOS にどのように対応するかが 1 つの課題だった．

また悪戯での虚偽登録を防ぎつつ陽性登録を確実に行ってもらうためには，保健所との連携が不可欠であった．個別自治体と連携して接触確認アプリを導入してもらうのか，国に後押ししてもらうのか，いずれにしても保健所による協力を得ようとすると，コミュニティだけで完結できるものではないという点が議論となった．

4 月に入るとコロナテックチームが立ち上がり，内閣官房コロナ室と民間との連携へ向けて協力関係の枠組みができた．そこで Code for Japan から接触確認アプリを提供する提案が行われた．当時，日本国内では 3 グループが接触確認アプリの構築に関心を持っていた． Code for Japan と，Covid-19 Radar，楽天の 3 者である．

当初コロナテックチームでは，陽性者情報の配信のみを国が構築し，民間が複数アプリを提供することを認めることも検討されていた．ところが Apple と Google の提供する Exposure Notification API を利用する場合には，公衆衛生当局が指定した 1 つのパブリッシャーに対してしか API の利用を認めないとする制限があった．

また有識者会議からは接触確認アプリのために陽性者の情報を配信する場合，そのデータコントローラは国であるべきとの整理が行われた．そこでコロナテックチームの事務方では当初，国から委託を受けた民間コミュニティがバックエンドを運用して，複数グループがアプリを提供することを想定していたが，バックエンドとアプリともに，厚生労働省が調達する方向に方針転換が行われた．この方針は

5月8日のコロナテックチーム会合にて発表されて，同様の内容は5月9日の有識者検討会合でも報告された．

なぜ Exposure Notification API を採用したのか

3月から4月にかけて，日本国内の各グループは独自の仕様で接触確認アプリを構築していた．当時はシンガポールの Trace Together もソースコードが開示されておらず，欧州7カ国が推す PEPP-PT（Pan-European Privacy-Preserving Proximity Tracing）と，スイスを中心としたグループが開発した DP-3T（Decentralized Privacy-Preserving Proximity Tracing）との間で主導権争いが行われていた．状況が流動的なことから各グループとも標準化の決着を待たず，独自方式での接触確認アプリの構築を模索していた．そんな中で4月10日 Apple と Google が DP-3T をベースとした Privacy-Preserving Contact Tracing（後に Exposure Notification API と改称）を発表した．

各グループとも5月ごろのアプリ公開を目標に置いており，Apple と Google からの API 提供を待っていてはリリースが遅れてしまう懸念があった．一方で日本国内では Apple の iOS のシェアが過半を超えており，iOS ではバックグラウンドで動作するアプリに Bluetooth 通信を認めていなかったことから，アプリを立ち上げ続けていない限り接触確認を行えないことが懸念として残っていた．

まずは独自方式でリリースを行い，後から Exposure Notification API に対応するのか，それとも当初から Exposure Notification API に対応するのかが論点としてあった．PEPP-PT の導入を推進する欧州委員会は Apple に対して，iOS アプリのバックグラウンド通信に対する制限を緩和するよう申し入れを行ったが，前向きな返答は得られていなかった．

リリース時期を優先するならば独自方式で変更させるべきであったが，iPhone で電源を入れてアプリを立ち上げ続けていなければ接触確認アプリとして動作しないとなると混乱を招く恐れがあった．独自方式から Exposure Notification API への移行が可能なのか技術的に詰められなかったことに加えて，Exposure Notification API 技術の詳細を調べる過程で，各グループが当初検討していた独自方式と比べると，かなりプライバシー水準が高いことが明らかとなった．

国際的に Exposure Notification API と同水準のプライバシーが求められるのだとすると，独自方式のプライバシー水準について国際的に非難を浴びることが懸念された．特に iPhone のシェアが高い日本において iOS で確実に動作することが非常に重要であること，利用者やプライバシー専門家・技術者コミュニティからの理解を得るためには，国際的に認められた十分なプライバシー水準を確保する必要があることから，独自方式は見送って当初から Exposure Notification API を利用することとした．

当初から Exposure Notification API を利用することとしたことによって諦めざるを得ない要件も少なからずあった．3グループからそれぞれアプリを出すことは，Apple Google 両社の API 利用規約で認められていなかったことから断念せざるを得なかった．また当初の仕様では陽性患者以外でもすれ違った人の数を記録し数え上げることによって，人の密集したところにできるだけ行かないように行動変容を促すことを考えていたが，この仕様は実現できなかった．また積極的疫学調査においては誰と接触したのかが重要となるが，この情報を接触履歴から取得することも断念せざるを得なかった．

プライバシーを優先して断念した動作の追跡

接触確認アプリの構築にあたっては，国民のプライバシーについて懸念があったことから，有識者会

議を設置して議論を行った．3 月頃からの日本でのコミュニティによる接触確認アプリ開発の機運を受けて，4 月初旬から各国の接触確認アプリ事情について調査していた IT 専門家，感染症専門家，プライバシー専門家，弁護士，憲法学者，政策担当者らが手弁当の有志で意見交換を行っていたところ，政府の方針を受けてコロナテックチームの下に有識者会議として改めて 5 月 9 日付で組織された．

この会議の中で，アプリケーションのデバッグのための情報収集や，ダウンロード数や陽性通知数といった計数をモニタリングすべきかについても議論が行われた．モニタリングの必要性は理解されたものの，プライバシー保護の観点から，その方式については Exposure Notification API に準じるプライバシー水準とすべきではないかと問題提起があった．

たとえばダウンロード数を正確に取るためには，ダブルカウントを防ぐために一意の識別子が必要となる．接触確認アプリにおいては端末が生成した日次鍵（Temporary Exposure Key）と，その日次鍵から導出された 10 分おきの接触符号（Rolling Proximity Identifier）が用いられるが，そういった技術方式を用いてプライバシーを保護していると国民に説明する一方で，不変の一意識別子を用いてアプリケーションの動作を監視することは，結果として利用者を騙したことになってしまわないか，といったことが懸念された．

アプリケーションの利用状況や不具合を把握するための製品の多くが一意の識別子を情報として含んでいることから，これらも利用しないこととした．接触確認アプリの機能の大半は OS 上で構築された機能であるため，アプリそのものの不具合を追跡しなくとも，大きな問題は起こらないのではないかという甘い見通しもあった．結果としてアプリの不具合に早い段階で気付いて障害を切り分ける手段は失われた．

こうした議論を踏まえ，5 月 26 日にテックチームからアプリ仕様書とプライバシーの評価を公表した．翌 27 日に厚生労働省は HER-SYS を委託していたパーソルプロセス＆テクノロジー（株）に接触確認アプリの開発および 7/31 までの運用保守を委託（HER-SYS の開発・運用保守に係る契約の追加契約），（株）エムティーアイ，日本マイクロソフト（株），（株）FIXER，イー･ガーディアン（株），ディザイアード（株）への再委託を承認した．

開発を運用保守業者に引き継いだ後，どうしても切り分けが難しい複数の不具合に直面したことから，9 月に改めて有識者検討会合を開催し，ログ情報蓄積・送信に関する仕様について諮った．プライバシー専門家からの意見を踏まえて，詳細仕様が明らかとはいえないアプリケーション解析用の製品を用いるのではなく，プログラム中で明示的にアプリケーション・ログを出力し，明示的に「ログを送信する」ボタンを押した場合のみ，利用者が送信前にログの内容を確認した上でメールにて送信する仕組みとした．この仕組みは 12 月 3 日にリリースされたバージョン 1.2.0 から実装されて，後述するバージョン 1.1.4 以降で発生した不具合の切り分けにも役立った．

再委託先事業者の選定経緯

前述した通り 5 月の連休明けには接触確認アプリについて，当初から Exposure Notification API を利用して，厚生労働省が調達する方針が明らかとなった．本来であれば構築を決めた時点で予算要求から一連の調達手続きを行う必要がある．

ところが第二次補正予算として執行する場合，突貫で二次補正の予算要求を行って予算の成立後に調達手続きに入ったとして，6 月中の予算成立が想定されていたことから，リリースは 7 月以降となることが想定された．各グループとも独自方式であれば 5 月中，Exposure Notification API に移行した場合であっても 6 月にはリリースできるとしていた．政治的にも緊急事態宣言後の感染防止対策として期待

されていたことから，できれば 5 月の緊急事態宣言明け，遅くとも 6 月にはリリースすることが望まれていた．

二次補正予算の成立を待たずに接触確認アプリを構築する場合，すでに確保している予算から執行する必要があった．新たに予算要求から調達手続きを実施するためには管理職級を充てる必要があったが，当時は厚生労働省・コロナテックチームともに管理職級の人材は払底していた．厚生労働省はすでに HER-SYS を開発しており，COCOA が保健所と連動する場合，HER-SYS との連携が想定されていたことから，HER-SYS の執行残を使って COCOA を構築することは，限られた要員でアプリを早期にリリースするためには最短の方法だった．

COCOA の開発を HER-SYS に含めるための契約変更にあたっては，数事業者が手を挙げていた事情を踏まえるならば，再委託先の選定にあたって，条件を揃えて相見積もりを取るといった方法も考えられた．主契約事業者が再委託先を選定することは広く一般に行われており，決して珍しいことではない．

接触確認アプリ COCOA に限らず，各府省の新型コロナ感染症対策のシステム調達では開発着手までの時間的猶予がなく，WTO ルールで定められた国際競争入札の手続きを踏むことが難しい．どうしても緊急随契，特命随契，既存契約の変更などで対応せざるを得なかった．

新規調達に要する期間と事務量が大きいために，既存案件の契約変更として新規の開発が行われ，結果として追加調達分の業者選定にあたって十分な競争性が担保されにくいことは，COCOA に限らず広く政府の IT 調達が抱えている課題であり，今後の改善が望まれる（**表 -1**）．

表 -1　令和 2 年度予算の閣議決定・成立のスケジュール

	R2 当初予算案	R2 一次補正	R2 二次補正
閣議決定	令和元年 12 月 20 日	令和 2 年 4 月 7 日	令和 2 年 5 月 27 日
成立	令和 2 年 3 月 27 日	令和 2 年 4 月 20 日	令和 2 年 6 月 12 日

初期リリース段階でのテストは十分だったか

5 月 25 日に総理が，個人情報はまったく取得しない，安心して使えるアプリを，来月中旬を目処に導入する予定である旨を発表した．6 月に入って開発が進むにつれて，どの段階でリリースを判断するかが課題となった．6 月中旬を目指していたが，6 月に入ってから画面設計書は示されたものの，テスト仕様書，テスト報告書などが示される気配はなかった．

ようやく 6 月 10 日に開発・テスト状況について数ページの資料が示されたが，その資料によると結合テストの一部しか行われておらず，詳細なテスト仕様書とテスト報告書が作成されていないように見受けられた．

そもそも 5 月 27 日に契約して 2 週間ほどしか経っていない段階で，一般的なシステム調達で求められるテスト仕様書・テスト報告書などをすべて求めることは時間的に無理がある．納品物にこだわって開発者に過度な負担をかけるよりも，走りながら状況を把握し改善する方針とせざるを得なかった．陽性登録のために必要な一連の処理フローを確認するためには HER-SYS 側で接続検証環境を払い出してもらう必要があったが，その目処も立っていなかった．

情報システムのプロジェクト管理における常識に従えば，リリースを延期すべきことは明らかだった．問題は仮にリリースを延期したとしても，テストのための環境や体制を整えることについてまったく見通しが立たないことである．仮に 2 週間から 1 カ月近くリリースを延期したところで，事態が好転する見込みはなかった．むしろ主要なオープンソース開発者が関与できる 6 月中に，できるだけ多くの課題を発見し，潰しておく必要があった．このままテスト環境が整うのを待っていたら，OSS 開発コミュニティがプロジェクト体制から外れて，まったく身

動きが取れなくなってしまうことが懸念された．

やむを得ず本番環境でテストするほかないということであれば，対象を限定してリリースすることが望ましい．そこでプレビューとして限られた方向けにリリースしてはどうかとする提案も行われたが，総理が 6 月中旬にリリースすると発言していたことと齟齬が生じてしまう．どのみち実際にアプリをリリースしてから，実際に数多くダウンロードされて利用者が陽性登録を行うまでには時間差がある．そこで 6 月 19 日の段階で対象は限定せず，広く一般にリリースすることとなった．一方で品質を担保できる見通しが立たないとする懸念は共有されて，まず 1 カ月近くはプレビュー版として提供し，広く利用者からのフィードバックを受けることとした．

残念ながらリリース直後から数多くの不具合が見つかった．そもそも HER-SYS との連携が動作せず，陽性登録を行うことができなかった．接触判定の重み付けについてダミーの値がそのまま登録されていて，仮に陽性登録できていた場合には，過剰に接触通知を出してしまうところだった．ほかにも有識者検討会合で認識されていなかったアプリケーション解析用の製品が組み込まれていたことや，軽微な文言修正やリンク切れの修正など，リリースから短期間で非常に多くの修正項目が見つかった．見つかったバグの性質としては，結合テストの途中段階の品質のままリリースされてしまったといえる．これらの仕様不整合とバグの一部は 7 月中旬にかけて，OSS 開発コミュニティを中心に改修を実施した上で，運用保守業者に引き渡された．

9 月の Android 版リリースにおける不具合の発生要因

COCOA バージョン 1.1.3 までは，陽性者との接触が確認された旨のプッシュ通知が表示されるものの，COCOA を開いて陽性者との接触を確認すると「陽性者との接触は確認されませんでした」と表示される不具合があった．これは接触リスク判定に関係なく，検知したすべての接触に反応してしまうことに起因していた．この結果 8 月から 9 月にかけて軽微な検知についても接触通知が多発してしまった．

アプリで表示するために具体的な接触内容を要約する処理においても iOS と Android とで Exposure Notification API の実装に差異があり，iOS では開発者が指定した閾値を超えた接触のみ処理するのに対して，Android では閾値に達していない分も含めた接触者すべての詳細情報を取得していたため，OS の通知だけでなくアプリ内でも過剰検知を行ってしまった．しかし，ユーザからのヘルプデスクへの問合せについても上述のプッシュ通知と画面表示の差異が発生していたのは iOS のみであったため，開発チームでは本不具合が発生しているのは iOS のみと受け止められていた．プライバシー配慮等の兼ね合いもあり，ユーザにどの程度接触通知が発生しているのかを開発者側が把握することができておらず，そもそもの通知が過剰であるかどうかを把握することは難しい状況であった．係るバグを修正するために，閾値と減衰リスクパラメータの値を見直した上で，検知した接触の最大リスク値が閾値よりも大きい場合だけ，通知とアプリ内の表示で接触について表示するための改修を行った．ところがこの際に，iOS と Android のリスク計算における感染リスクパラメータの取扱いの違いを把握できておらず，iOS では適切に検知するが，Android ではまったく接触通知を行わなくなってしまった（**表 -2**，**表 -3**）．

Exposure Notification API では，検知した個々

表 -2　バージョン 1.1.3 までのアプリの挙動

	OS 通知	アプリ内
iOS	**過剰検知**	仕様通り
Android	**過剰検知**	**過剰検知**

表 -3　バージョン 1.1.4 のアプリでの挙動

	OS 通知	アプリ内
iOS	仕様通り	仕様通り
Android	**検知せず**	**検知せず**

の接触について，4種のリスクパラメータについて，0から8までのレベルを記録し，それぞれのレベルに対応して配列で定義されたスコアを引き当て，乗算することによってリスク値を算出し，その数字が閾値を超えているかどうかで通知すべき濃厚接触かどうかを判定する．

COCOAではリスク計算の設定として感染リスクスコア配列に7,7,7,7,7,7,7,7を代入し，個々の接触について感染リスクレベルとして0を代入した．実際に陽性者との接触があった場合，感染リスクスコア[0]に当たる7が代入されることを想定していた．しかしGoogleの実装では感染リスクレベル1に対して感染リスクスコア[0]が代入され，感染リスクレベル0に対しては固定値1を返す実装となっていた．

このことは文書内で特に記載されておらず，Googleが1.1.4リリース後の9月末に公開したExposure Notification APIの内部実装のソースコードを読まない限り把握できなかった（**図-1，表-4～6**）．

実機テストの実施を阻んだ開発環境の制限

COCOAで直面したiOSとAndroidにおけるExposure Notification APIの実装差異は技術文書で明確に記載されていなかったこと，5月から6月にかけての開発段階では技術文書を補完するソースコードが公開されていなかったことから，事前に予測して回避することは困難だった．しかしながら実機テストを行っていれば，容易に検出できたはずである．なぜCOCOA 1.1.4のリリース時に，十分に実機テストを行えなかったのだろうか．

COCOAは7月まではCovid-19Radar開発チームが関与するかたちで開発が行われた．8月から運用保守業者が開発を引き継ぎ，9月に入ってバージョン1.1.3，1.1.4と相次いでリリースしている．引き継がれた資材の中にはアプリとバックエンドについて，リリース版と開発版のビルドパイプラインが含まれていたが，バックエンドが動作するためにはHER-SYSとの連携が必要だった．リリース版は本番系のHER-SYSと接続できたが，開発版が接続できるHER-SYSの接続検証環境がなかったため，偽の陽性登録を行って実際に接触検知が行われるかどうか確認することができなかった．

COCOAはHER-SYSの一部として開発されていたが，HER-SYS自体の開発に用いるための改修確認環境はあっても，HER-SYSと接続してCOCOAの動作検証を行うための接続検証環境が提供されなかった．このことは当初リリース時にHER-SYSとの接続が動作しなかったときから露見していたが，本稿執筆時点も改善を確認できていない．

接触確認アプリの性質上，本番システムで偽の陽性登録を行うと，周囲の端末が陽性者との接触として検知してしまう．そのため一般の利用者に影響を与えないように実機テストを実施するためには，電波暗室などを用いて電磁的に隔離した環境でテストを実施するか，本番系・開発系とは別にテスト用の診断キーを配信する専用の環境を構築してテストを行う，またはテスト用の診断キーを簡単に登録できるテスト専用のアプリを構築するなどの作業が必要だった．

Exposure Notification APIは短期間で開発され，頻繁に更新され，それぞれの地域で公衆衛生当局が認めた機関しか利用できないため，入手できる技術情報が非常に限られている．それぞれのパラメータの重み付けに対して実際どのような挙動を示すかについて，仕様書の行間を勝手に解釈するのではなく，十分な検証を行う必要があった．

HER-SYSから接続検証環境が提供されないのであれば，HER-SYSのように振る舞うスタブまたは実機検証用のバックエンドを構築するか，簡単にテスト用の診断キーを登録できる，テスト専用のアプリを構築する必要があった．実際，スイスを中心に開発したDP-3Tでは，接触確認アプリとは別に

Exposure Notification API を評価するための簡単なテスト用アプリを構築している．この実機テスト環境の構築は，本来なら接触確認アプリの感度に影響するパラメータを変更するバージョン 1.1.4 のリリースよりも前に実施すべきだった．

委託開発業者は 10 月 12 日には HER-SYS を模擬するスタブを開発し，実機テストを行える環境を構築したが，9 月のリリースに遡って実機テストを

```
int transmissionRiskScore =
  diagnosisKey.getTransmissionRiskLevel() == RiskLevel.RISK_LEVEL_INVALID
          ? 1
          : configuration.getTransmissionRiskScores()
            [bucketRiskLevel(diagnosisKey)];
```

https://github.com/google/exposure-notifications-internals/blob/main/exposurenotification/src/main/java/com/google/samples/exposurenotification/matching/RiskScoreCalculator.java#L69-L72 より引用

図 -1　Google が 9 月末に公表した Exposure Notification API のソースコードの一部

表 -4　iOS, Android における感染リスクレベルの解釈の違い -

感染リスクレベル		使途				代入されるリスク値	
設定	値	Apple	Apple 推奨	Google 推奨	Google ※	Apple	Google
当初	**0**	**任意**	**未定義 / 任意**	**未使用**	**無効**	**スコア [0]**	**固定値 1**
未使用	1	任意	確定検査　リスク　低		リスク最低	スコア [1]	スコア [0]
未使用	2	任意	確定検査　リスク　標準		リスク低	スコア [2]	スコア [1]
未使用	3	任意	確定検査　リスク　高		リスク低中	スコア [3]	スコア [2]
修正後	**4**	**任意**	**確定診断**		**リスク中**	**スコア [4]**	**スコア [3]**
未使用	5	任意	自己申告		リスク中高	スコア [5]	スコア [4]
未使用	6	任意	陰性		リスク高	スコア [6]	スコア [5]
未使用	7	任意	再発		リスク超高	スコア [7]	スコア [6]
未使用	8			未定義／任意	リスク最高		スコア [7]

それぞれ Apple☆1 Apple 推奨☆2 Google 推奨☆3 Google☆4 を元に作成

表 -5　各バージョンにおける設定値

版数	閾値	感染リスク	接触期間	経過日数	減衰リスク
1.1.2 ※	1	1,2,3,4,5,6,7,8	1,2,3,4,5,6,7,8	1,2,3,4,5,6,7,8	1,2,3,4,5,6,7,8
1.1.3	1	7,7,7,7,7,7,7,7	0,0,0,0,1,1,1,1	1,1,1,1,1,1,1,1	0,0,0,0,1,1,1,1
1.1.4 以降	21	7,7,7,7,7,7,7,7	0,0,0,0,1,1,1,1	1,1,1,1,1,1,1,1	1,2,3,4,5,6,7,8

※ 表の記載は GitHub 上で確認できるハードコーディングされた規定値．実際には CDN からダウンロードされた JSON ファイルの設定値が優先して適用され，6 月 24 日からバージョン 1.1.3 相当の設定値で運用されていた．

表 -6　バージョン 1.1.4 のアプリでの挙動

感染リスク	接触期間	経過日数	減衰
1	0	**1**	1
7	0	**1**	2
7	0	**1**	**3**
7	0	**1**	**4**
7	**1**	**1**	**5**
7	**1**	**1**	**6**
7	**1**	**1**	**7**
7	**1**	**1**	**8**

固定値 1 に対して 3 ～ 8 をかけても **21** を超えない

固定値 7 に対して 3 ～ 8 をかけると **21** を超える

iOS の場合
7 × 1 × 1 × 3 = 21 閾値に達するので通知を行う
Android の場合
1 × 1 × 1 × 3 = 3 閾値に達しないため通知を行わない

☆1 https://developer.apple.com/documentation/exposurenotification/enexposureconfiguration/exposure_risk_value_calculation_in_exposurenotification_version_1
☆2 https://developer.apple.com/documentation/exposurenotification/enexposureconfiguration/3586323-transmissionrisklevelvalues
☆3 https://developers.google.com/android/exposure-notifications/exposure-notifications-api
☆4 https://github.com/google/exposure-notifications-internals/blob/main/exposurenotification/src/main/java/com/google/samples/exposurenotification/nearby/RiskLevel.java#L38-L49

行うことはしなかった．12 月にはバージョン 1.2.0 をリリースしたが，本人同意に基づいて動作情報を送信する機能の追加で，感度などに影響する改修は行わなかったことから，実機テストにおいて，接触通知の網羅的なパターン検証までは行わなかった．

10 月の段階で実機テストの環境を構築できており，11 月には GitHub 上の issue として Android 版で通知が行われない旨について報告がなされていたにもかかわらず，なぜ 12 月のリリースで実機テストを十分に行わなかったのだろうか．

GitHub 上の issue が見落とされた背景

COCOA の元となった Covid-19Radar のリポジトリは COCOA のリリース前から，COCOA のソースコードは 9 月から GitHub 上で公開されているが，実際の開発が GitHub 上で行われているわけではない．実際の開発ブランチは Azure DevOps 上で運用管理され，リリースのたびに GitHub 上で公開されている．

これは元となった Covid-19Radar のライセンスである Mozilla Public License を遵守する必要があること，各国の接触確認アプリが同様にソースコードを公開していることを踏まえて，受託ベンダに対してソースコードの GitHub での公開を依頼した．しかしながら GitHub 上での issue や Pull Request の取り扱いについて，運用保守契約の中で適切な対処方針と SLA（Service Level Agreement）を規定することまでは難しかった．

契約上，委託業者が GitHub 上の issue を確認する義務はなかった．指摘の多くは技術的なもので，厚生労働省の職員が内容を理解することは難しかった．技術的スキルの面では担当する政府 CIO 補佐官が対処できる可能性も考えられたが，募集時の職務内容からは大幅に逸脱している．そもそも現行の IT 調達における前提としては，ソースコードを公開して，多くの技術者から技術的な指摘を受けることは想定されておらず，そのような体制は組まれておらず，手順や役割分担についても定まっていないのが実情である．

ソースコードの公開場所として GitHub を選んだことで，issue や Pull Request の取り扱いについて開発コミュニティから過大な期待を招いてしまった一方で，issue や Pull Request に誰が責任を持って対応するか事前に決めていなかったことで，せっかく重大な不具合について指摘をいただいたにもかかわらず適切な対応ができなかった．

一般的なシステム調達においては，開発ベンダが瑕疵担保責任を負っており，不具合が見つかった場合には改修する責任を負う．その場合だと瑕疵に該当する issue が GitHub 上で上がった場合には，自発的に対応することも考えられる．しかしながら COCOA の場合，オープンソースコミュニティが開発したアプリを引き取って運用保守する形式となっていたため，開発元であるオープンソースコミュニティとの間には契約がなく，運用保守を引き継いだ業者に対して瑕疵担保責任を問うことが難しかった．リリースへ向けた作業として，開発委託先に対してはテストや品質管理などの役割を期待していたが，リリース時に満たすべき品質水準を指示できておらず，十分な受入テストの体制もなかった．

結果としてオープンソースコミュニティ，開発委託先ともに積極的に GitHub の issue を管理して課題を解決するインセンティブが働きにくかった．発注者が主体的に GitHub の issue を管理し，課題管理表を更新して運用保守業者に対して作業指示を行う必要があったが，その必要性は認識されておらず，体制も整ってはいなかった．

オープンソースコミュニティでの指摘や改善提案は，一般的なソフトウェアの瑕疵担保責任と比べて事前に工数を読みづらいため，総額固定の請負契約にはそぐわない面がある．SLA を規定して委託業者に GitHub の管理を委ねるのであれば，工数に応じて委託費が支払われる単価契約や準委任契約とす

ることが望ましい．開発委託とは別に技術スキルを持った職員を雇い入れることも考えられる．現在COCOAではアプリ開発に詳しい専門家が厚生労働省参与として着任し，COCOAのGitHubの管理に携わっている．

厚生労働省の調査報告書と，今後の再発防止に向けて

2021年4月16日，厚生労働省はCOCOAについて不具合の発生経緯の調査と再発防止の検討についてとりまとめた．おおむね筆者の認識と齟齬はなく，本稿も当該報告書の記載を参考に執筆している．

この報告書は9月のバージョン1.1.4のリリースによってAndroid版で通知が行われなくなった事案を受けての調査だが，詳述してきたようにCOCOAについては調達制度，体制，業者選定，当初のリリース判断，その後のリカバリーと再発防止策，保守運用業者への引継など，さまざまな課題があり，結果として当該事案に繋がっている．

厚生労働省が再発防止策として挙げているように，開発当初から外部システムとの結合テストを実施するための環境を整備しておき改修のつど一連の動作検証を行うこと，連携先システムやAPI等の改修による影響を受けることを前提として，不確定要素を前提とした契約の在り方を検討すること，政策判断を担う管理職自身がITリテラシーを持って，外部有識者などのIT専門家を活用していくことは再発防止に資すると考えられる．

では今後デジタル庁が創設されて，IT専門家が配置された暁には，COCOAのような問題の再発を防ぐことができるのだろうか．外部からITに精通した人材を管理職や技術者として登用することによって，いくつかの問題が改善することは期待される．しかしながら十分な品質のサービスを提供できる体制を整備するためには，ほかにも改善すべき点が多い．

オープンソースや外部サービスを活用して迅速にサービスを構築し，責任を持って提供するためには，発注者が主体的に品質に対する責任を負うことが求められる．納期・費用・品質がトレードオフの関係になりがちな中で，政治的に費用と納期がピン留めされてしまうと，品質に皺寄せが行ってしまいがちだ．

ベンダ任せにせず，主体的に品質管理の責任を負うためには，予算内でのスケジュールの短縮圧力がかかりがちなプロジェクト責任者からは独立して，品質に対して責任を負う役割を置き，品質管理やテストの専門家がサポートすることが望ましい．現状では受入テストも形骸化しており，情報システムの品質管理は実質的にベンダ任せとなってしまっているケースが少なくない．

オープンソースコミュニティや外部サービスとの連携を通じて，幅広い利用者に対して使い勝手が良く安心して使えるスマホアプリを迅速に提供することは，今後のディジタルサービスでCOCOAに限らず広く求められる．こうした不確定要素を孕んだシステム開発を推進するためには制度・体制ともに見直しを要する．COCOAの失敗を関係者のITリテラシーの欠如と片付けてしまうのではなく，硬直的な調達制度や要員管理，管理態勢にメスを入れて，激しい環境変化に柔軟に適応できる開発体制と，品質管理に必要なガバナンスを確立する端緒とすべきではないか．

（2021年4月30日受付，2021年5月15日note掲載版）

楠　正憲（正会員）　masanork@gmail.com
マイクロソフト，ヤフーなどを経て2017年からJapan Digital Design CTO．内閣官房 政府CIO補佐官としてマイナンバー制度を支える情報システム等の構築に従事．接触確認アプリに関する有識者検討会合構成員，厚生労働省CIO補佐官（2020年5月～11月），内閣官房IT総合戦略室と厚生労働省の連携チームとして接触確認アプリCOCOAの開発・運用に関与．