ClickOnce×WCF+XML＝Web アプリと Client アプリの融合

（株）あいおい保険システムズ

■ 執筆者Ｐｒｏｆｉｌｅ ■

吉原 怜

2002 年　ビーコンシステム（株）入社
2007 年　同退社
2007 年　（株）あいおい保険システムズ 入社
　　損調プロジェクト担当

■ 論文要旨 ■

クライアントアプリケーション時代から Web アプリケーション時代、モバイル時代とアプリケーションのトレンドが変化してきた(技術隆替と呼ぶ)。技術隆替により、企業によっては Web アプリとクライアントアプリが混在している。また、今回開発を行った「見積管理システム」（以下、今システムという)では、費用対効果の観点からノート端末に通信機器の採用が見送られ、Web アプリとクライアントアプリが混在することとなった。この混在はシステム側の問題がユーザーの作業効率の低下をもたらしたという悪い例である。

ClickOnce と WCF、XML を使用することで技術隆替の果てに混在してしまった Web アプリとクライアントアプリを融合することを可能とし、システムを本来の「ユーザーをサポートする」姿に戻すことに成功した。

本論文では、今システムでの事例・提案を通して Web アプリとクライアントアプリを融合させる方法を紹介する。

■ 論文目次 ■

■ 図表一覧 ■

1　はじめに

1．1　当社の概要

当社は 2001 年 7 月、旧千代田火災システムズエンジニアリングと旧大東京システム開発の統合によって発足しました。現在、保険・金融業界は高品質な商品・サービスの提供が求められている。そのために高度な業務処理を支える IT 機能が不可欠となっており、もちろんあいおい損保グループも同様である。当社はあいおい損保グループのシステム部門の中核として、合理的･効率的なシステムの企画・開発・保守・運用を行っている。

1．2　隆替のアプリケーション

1.2.1　クライアントアプリの時代

90 年代後半に隆盛した企業クライアントアプリ(クライアント／サーバ型システム(Windows アプリケーション))の最大の難点は端末へのリリースである。ユーザー自身にアプリケーション(以下アプリという)の更新を実施してもらう運用では、ユーザーの数が増えれば増えるほど問い合わせやリリース適用漏れが発生しユーザー側・システム側の双方に負荷が増大する。こうして多くの配信テクノロジーが乱立し、システム開発者はそれぞれの配信技術を習得し、それにあわせた納品を行うなど、負荷が増加していった。

1.2.2　Web アプリの時代

90 年代の反省を踏まえて、00 年代前半には各ユーザー端末へのアプリのリリースが不要な、Web アプリの採用が飛躍的に増加した。これによりシステム開発者のリリース負荷は軽減されることとなったが、Web アプリには以下の大きな問題があったのである。

・ クライアントアプリよりも操作性や応答速度が貧弱である。
・ Web アプリと既存のクライアントアプリを連動させることができない。

Web アプリの採用はシステム開発者から見ると大きな効率化につながるが、ユーザ側から見ると業務効率化の後退を意味するのである。

1.2.3　通信技術の時代

00 年代後半の現在では通信技術の発達により、携帯電話での業務入力や、通信機器を介してのノート端末での業務入力が増加している。しかし、文字入力の効率や、既存アプリとの連携、通信コストの面から見送られるケースも少なくない。今システムも同様に費用対効果の観点から通信機器の使用は見送られることとなった。

1.2.4　技術隆替の果てに

上記のとおりアプリには時代によって流行り・廃りが発生している。私はこれを「技術隆替(ぎじゅつりゅうたい)」と呼ぶ。技術隆替により企業によってはクライアントアプリと Web アプリが混在している状況である。また、今システムのような通信機器を使用しないノート端末を使用する場合でもクライアントアプリと Web アプリが混在してしまうのである。

技術隆替がもたらしたのはアプリの混在というシステム側の問題ではなく、ユーザー業

務への悪影響である。社外業務など一部ではクライアントアプリが動作する必要がある。だが社内業務の中心は Web アプリに移行しており、Web アプリとクライアントアプリの間では連携が取れないというジレンマを抱えている。ユーザーが意識して相互にアプリを動作させなくてはならず、作業効率の低下を招いている。つまり、システム側の問題解決の積み重ねによる影響をユーザーがカバーしているのである。これは「ユーザーをサポートするのがシステムの役割」という定義から反することである。

この論文では今システムでの事例を通して、技術隆替の果てに混在してしまった、Web アプリとクライアントアプリをユーザーの負荷が軽減するように連携させる方法を、ClickOnce と WCF、XML という技術を元に記載する。

１．３　ユーザー業務と規模

今システムを使用するユーザーは営業員である。営業員は社外で顧客と各種商談を行い、現地でデータを既存の別クライアントアプリと連携、入力することで算定などを行う。帰社後端末で更新した情報をサーバに戻し情報共有を行っている。

営業員(約 1000 人)は拠点(約 200 箇所)でデータを事前にダウンロード(以下 DL という)し端末に保存している。図１にユーザ業務とシステムの関わりを示す。

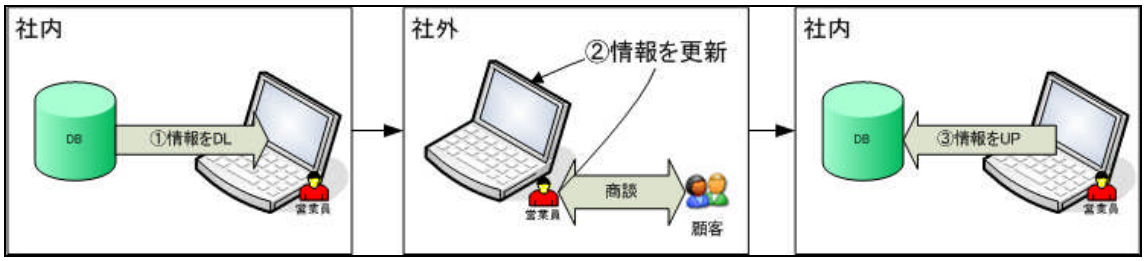

図 1　ユーザー業務

１．４　システム要望と制約

ユーザーからの要望は「リリースやバグ対応が迅速にできるので基本的には業務は Web アプリで行い、必要なデータのみ端末に DL し社外で登録を行う。同様の登録を社内で行うこともある。ただし、端末内にデータベース(以下 DB という)アプリを導入することはできない。また、年齢層が高くパソコンに不慣れな社員も業務を行う。」ということである。

考慮すべき制約は以下のとおりである。

- 登録業務は「社外(オフライン)」と「社内(オンライン)」の両方で行われる。
- 端末に DB のクライアントアプリを導入することはできない。
- 端末に DB アプリを導入することはできない。
- ユーザーに業務以外のシステム制約(エクスプローラーでファイル名を変換するなど)のお願いはできない。

２　　提案と反応

２．１　提案の前提条件

以下の２点を前提として考えることで、「端末 DB」の制約を解決し、提案の糸口とした。

・ 端末内のデータの保持は XML(Extensible Markup Language)を使用する。
・ 基本業務には ASP.NET または Java を使用した Web 型とする。

2.2 以降の提案箇所でのアプリの名称、役割を以下のとおりとする。
・ WebApp(業務 Web アプリ)
・ WebSrv(取得登録 Web サービス)
・ CLEntry.exe(登録処理クライアントアプリ)
・ CLGetter.exe(取得処理クライアントアプリ)
各アプリの役割は順次説明をする。

２．２ 提案１：XML を端末に手動 DL

2.2.1 提案

図２に提案１を示す。

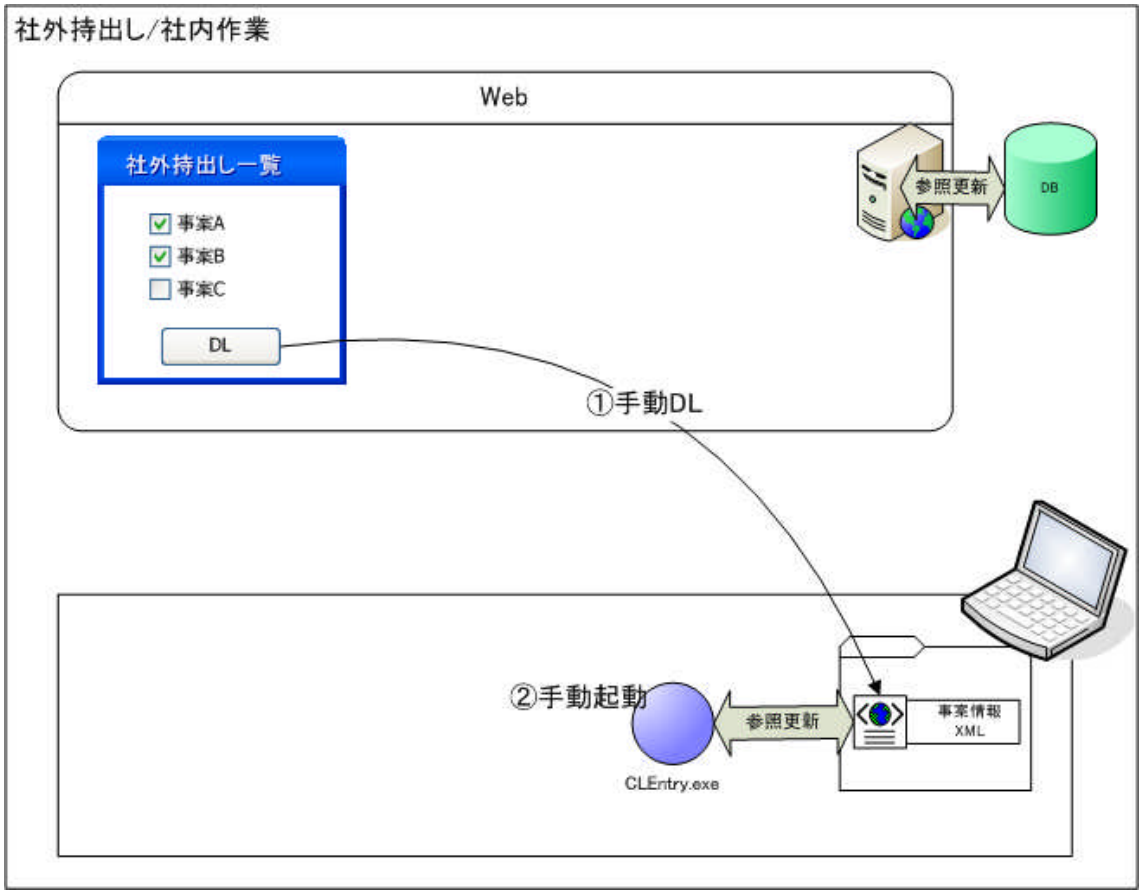

図 2 提案１：XML を端末に自動 DL

WebApp で作成した XML ファイルをユーザーに DL してもらう方法である。ユーザーの作業フローは次のようになる。

社外への持ち出しは WebApp で対象を選択し、ユーザーが指定した端末内フォルダに DL を行う(図２の①)。社内での登録においても同様の操作で実現をする。登録処理に関して

は CLEntry.exe で上記 DL データ(XML)を操作することで実現する。
　ユーザーにはファイル保存のダイアログを操作して指定箇所にファイルを保存してもらう必要があるが、決して高いハードルではないのではないかと考えられる。

2.2.2　ユーザーシステム部門の反応

- パソコンに不慣れな社員にはフォルダの選択などの DL 操作はハードルが高く厳しい。
- フォルダ選択をせずに指定箇所に自動で DL してほしい。
- 社内では、WebApp から直接編集を行えるようにしてほしい。

2.2.3　考察

ユーザーの指摘は以下の点において非常に厳しいものであった。

- ウィルス混入の恐れがあるので、一般的に Web アプリからユーザー端末の指定箇所にファイルを自動で DL することは出来ない。
- 登録処理を WebApp から直接編集となると、「社外での登録用の CLEntry.exe」と「社内での登録用 WebApp」の両方に登録処理を用意する必要が生じてしまい保守性の面から好ましくない。またユーザー業務としても同じ業務で異なる画面を操作することとなり、混乱を招くこととなる。

2．3　提案２：ClickOnce を利用して CLEntry.exe を起動し、XML を自動 DL

2.3.1　提案

図３に提案２を示す。

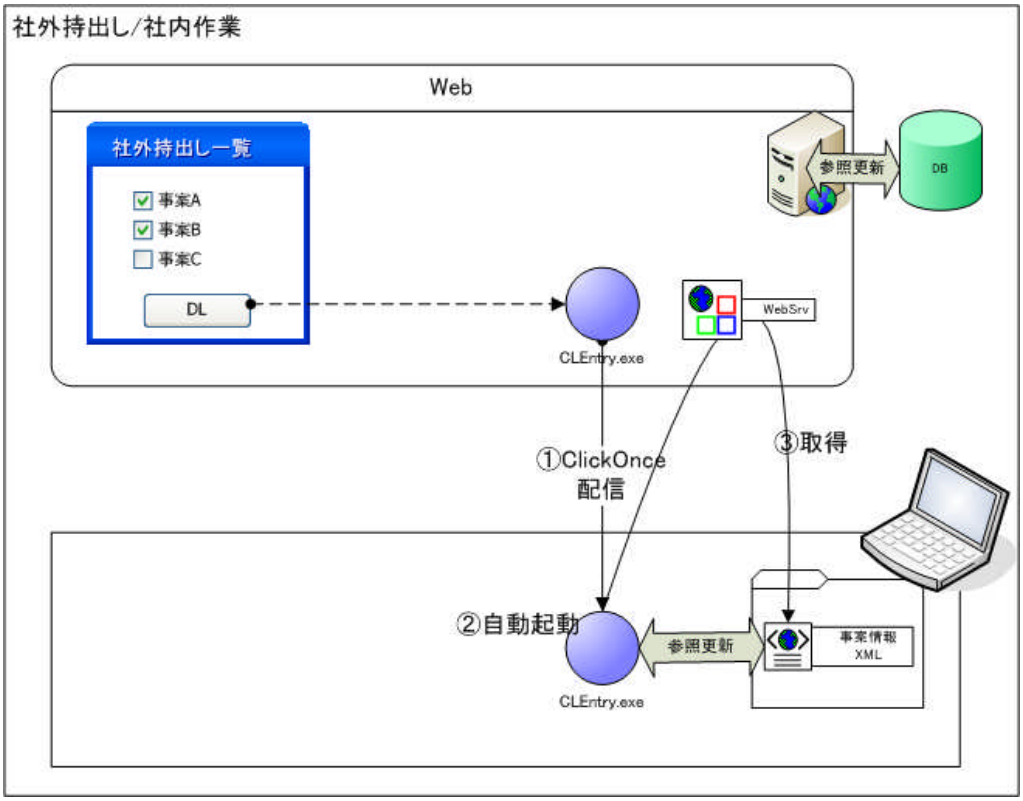

図 3　提案２：ClickOnce を利用して CLEntry.exe を起動し、XML を自動 DL

Microsoft が.NET で提唱している「ClickOnce」という端末へのアプリ配信のテクノロジーを利用する方式である。ClickOnce で配信されたアプリはサーバと通信することで、バージョンチェックを行いアプリを最新化した上で、起動する。配信されるのはクライアントアプリなので、端末内のファイル操作を行うことが可能となる。これにより指定箇所への自動 DL が可能となる。ユーザーの作業フローは次のようになる。

社外への持ち出しは WebApp で対象を選択し、ClickOnce で CLEntry.exe を起動し、WebSrv を利用して、XML を取得する(図３の①～③)。社内での登録においても同様の操作で実現をする。登録処理は ClickOnce で起動された CLEntry.exe のみで行う。

これでユーザーがすべきことは WebApp から対象を選択するだけとなり、負荷軽減となる。そして、社外と社内で同じ仕組みを使うことで保守性も維持でき、提案 1 での指摘事項はすべて解決することとなる。

2.3.2 ユーザーシステム部門の反応

- 社内業務でも DL 後に LAN ケーブルを外したら社外への持ち出しが可能となるなど、社内での登録時に XML を DL させるのはセキュリティとして許可できない。

2.3.3 考察

登録処理を「社外での登録用の CLEntry.exe」と「社内での登録用 WebApp」で分けるのは提案１と同じく保守性の面から導入できない。登録処理を社外でも社内でも同じ CLEntry.exe で行うことを前提に考えると、社外では DL した XML を DB と判断する必要があるが、社内では DB サーバを DB と判断する必要があり、取得更新処理を各 DB 用に 2 つ作成するなど保守性に大きな障害が生じる。

２．４　提案３：ClickOnce を利用してデータ取得(CLGetter.exe)を起動する

2.4.1　提案

図４に提案３を示す。

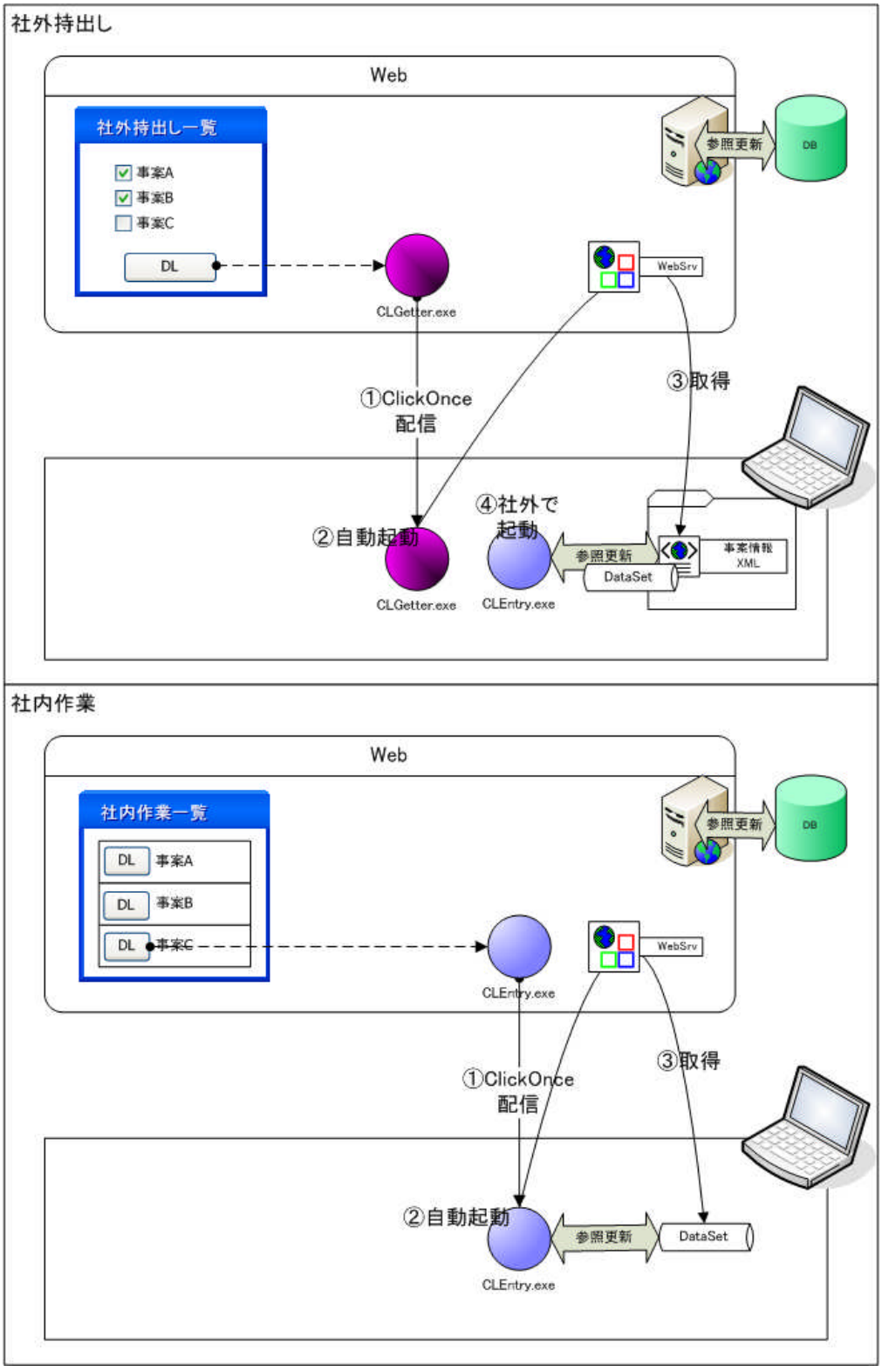

図 ４　提案３：ClickOnce を利用してデータ取得(CLGetter.exe)を起動

提案２では、データの取得と登録を CLEntry.exe のみで行おうとしていた。今提案では社外への持ち出し機能と登録機能を明確に分離することで、社内での XML の DL を排除しセキュリティを向上させることに成功した。

社外への持ち出しは WebApp で対象を選択し、ClickOnce で CLGetter.exe を起動し、WebSrv を利用して、XML を取得する(図４の①～③)。社内での登録においては、ClickOnce で直接 CLEntry.exe を起動する。

登録業務は社外・社内を問わず CLEntry.exe で行う。この際 CLEntry.exe は端末上に XML ファイルが存在する場合は XML ファイルからデータを取得する。XML ファイルが存在しなければ WebSrv を利用してデータを取得する。取得したデータはメモリ内部に展開し同じ DataSet という形式で持つことにより、CLEntry.exe は DataSet のみを DB と考えることが可能となり、提案２での問題が解消される。

2.4.2 ユーザーシステム部門の反応

セキュリティの加味、社外・社内でのユーザーの操作性の統一が図られており良いとの評価を受けた。ただ ClickOnce に関しては自社での実績がないので運用グループに確認を取ることとした。

2.4.3 運用グループの反応

- CLEntry.exe 起動時に毎回サーバにバージョンチェックを行うのは現実的ではない。
- CLEntry.exe と CLGetter.exe がリリースされるたびに、全ユーザーが DL するので回線の帯域を占有してしまい他システムへ影響が出てしまう可能性がある。

2.4.4 考察

ここまで ClickOnce ありきで考えており、WebApp からクライアントアプリである CLEntry.exe/CLGetter.exe を起動する別の方法を模索するには時間的にも技術的にも困難であると考えられた。ただ、運用グループの指摘は的確であり、他のシステムへの影響を考えると帯域を占有するような恐れのある技術は導入できない。

指摘を精査すると、自動配信での帯域の占有が一番大きな問題であることがわかる。つまり、帯域の占有を解消できれば ClickOnce を使用する方法でも可能なのである。

２．５　最終型：ClickOnce を利用してアプリを起動させるアプリを起動する

図５に最終型のシステム図を示す。

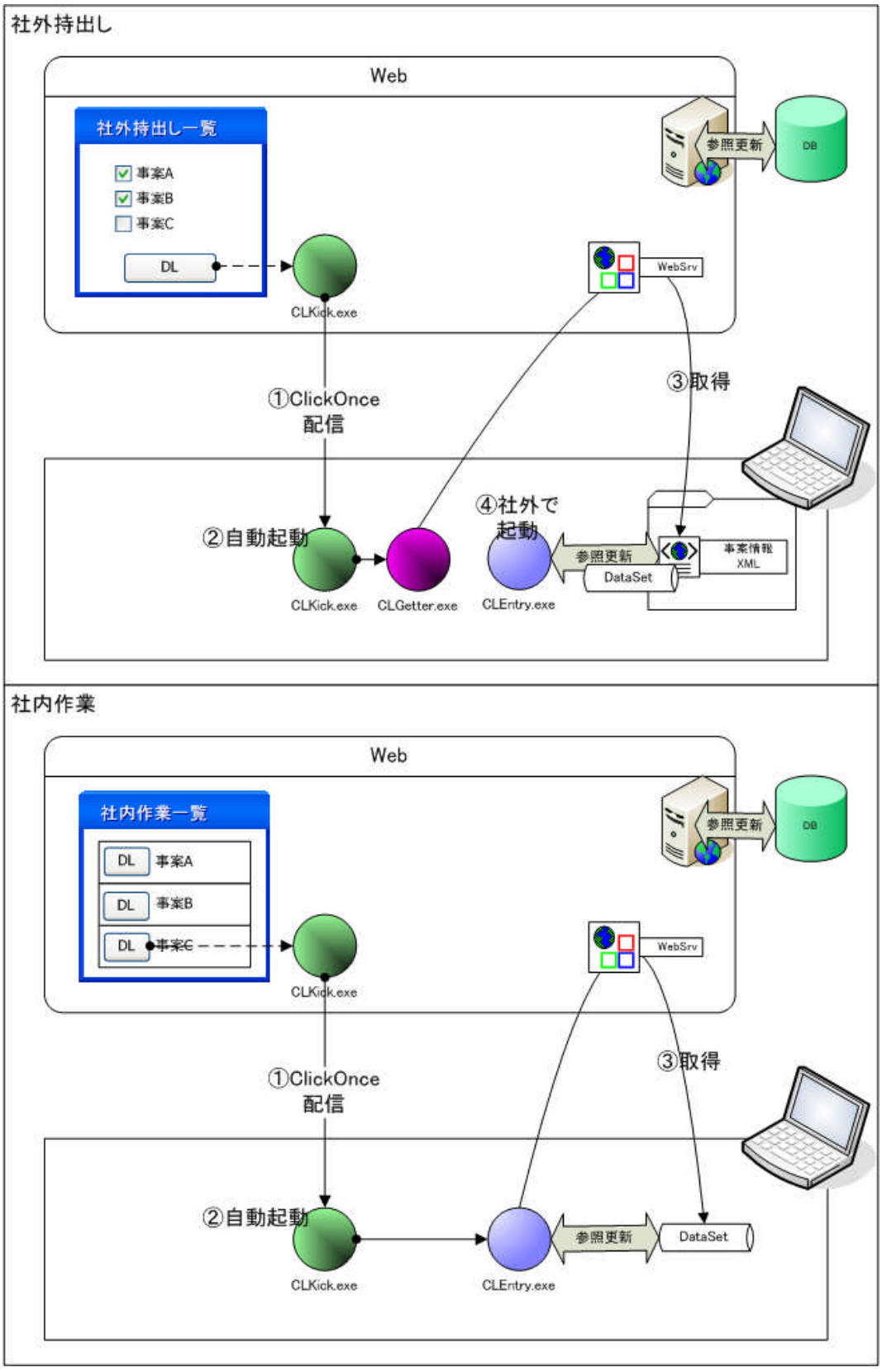

図 5　最終型：ClickOnce を利用してアプリを起動させるアプリを起動

CLEntry.exe/CLGetter.exe は CD などで各端末に事前にインストールを行う。同アプリのリリースは自社内での実績を加味し、SystemWalker を使用する。

ClickOnce は、各クライアントアプリ(CLEntry.exe と CLGetter.exe)を起動するだけのみに使用する(以下 CLKick.exe とする)。つまり、CLKick.exe を ClickOnce として配信・起動する。CLKick.exe は引数によって CLEntry.exe と CLGetter.exe を起動する。

CLKick.exe は Web から起動する必要のあるアプリの種類が増えない限り更新が発生しない。また、CLKick.exe はシンプルなアプリとなり容量も小さくできる。以上の点で、帯域の占有を抑えることができ、運用グループの指摘も解消することが可能となる。

社外への持ち出しは WebApp で対象を選択し、ClickOnce で CLKick.exe を起動し、CLKick.exe が CLGetter.exe を動かし WebSrv を利用して、XML を取得する(図５の①～③)。社外では CLEntry.exe を起動し、XML からの DataSet に対してデータの登録・更新を行う。

社内での登録においては、CLKick.exe で CLEntry.exe を起動し WebSrv を使用して、データをメモリ上 DataSet にコピーする。

上記のとおり、CLEntry.exe と CLGetter.exe の起動させるために CLKick.exe を用意しただけなので、ユーザーの業務としては、提案２・提案３と同じく WebApp から対象の事案を選ぶだけとなる。またシステム側から見ても、提案３と同じく、CLEntry.exe の DB は DataSet だけとなり保守性も確保できる。

また、CLKick.exe として分離したことで、既存の他クライアントアプリも Web アプリから CLKick.exe 経由で起動することが可能となる。これにより既存アプリも含めて１つの大きなアプリとなり、ユーザーに利便性をもたらすこととなった。

２．６ シームレス起動のための WCF

ユーザーに提供するに当たり、注力したのが WebApp から CLKick.exe 経由の CLEntry.exe の起動速度である。ユーザーの待ち時間がより少なくなるようにし、ストレスを軽減させなければいくら裏の仕組みが最新だとしてもユーザーの利益にならないからである。

ClickOnce に関しては、前項までに記載したとおり細分化を行いアプリ量を減らすことで軽減ができたのではないかと考える。

次に着目したのは WebSrv である。.NET テクノロジーにおける Web サービスでは、.NET Framework 1.0 の時代から「ASP.NET Web サービス(XML Web サービス)」が広く普及している。ただ、.NET Framework 1.0 は 2002 年登場の過去の技術であり、調査すると.NET Framework 3.0 からは「WCF(Windows Communication Fundation)サービス」というテクノロジーが誕生している。

Microsoft によると、『「WCF」 は 「ASP.NET Web サービス」 よりも 25 ～ 50%高速』と評価されており、実績はなかったがユーザーの利益を考え、WCF サービスを採用することとした。

３ おわりに

３．１ プロジェクトの結末

こうして自社では使用していない ClickOnce と WCF、.NET Framework 3.5 を根底に据え

て設計・開発に突入した。当然ながら ClickOnce と WCF で開発したことのある技術者を集めることはできず、メンバー全員が初挑戦となった。様々なトラブルに見舞われたが無事リリースできたのは、毎晩遅くまで挑戦を続けてくれたメンバーのお陰であり大変感謝している。

リリース後も、ClickOnce と WCF、XML を活用するという今回の仕組みが「使いづらい」というご指摘はないので、『No news is good news.（便りのないのは良い便り）』のことわざ通り、ユーザーにも満足いただけたと信じている。

参考までに、今プロジェクトで使用した技術及び製品・書籍などを以下に記載する。

- .NET Framework 3.5
- Microsoft Visual Studio 2008
- ASP.NET
- ClickOnce
- WCF(Windows Communication Fundation)
- Microsoft Enterprise Library
- AJAX(Asynchronous JavaScript + XML)
- XML(Extensible Markup Language)
- Active Reports＜帳票作成ソフト＞
- Excel Creator＜Excel 作成ソフト＞
- YUI Compressor＜javascript/css 圧縮ツール＞
- Sandcastle＜ソースからクラス仕様書を作成するツール＞
- BugTracker＜バグ管理ツール＞
- Systemwalker
- 『NET エンタープライズ Web アプリケーション開発技術大全(Vol.1～3)』(赤間 信幸)
- 『Microsoft Visual Studio 2005 による Web アプリケーション構築技法』(赤間 信幸)
- Patterns & Practices(http://msdn.microsoft.com/ja-jp/library/dd335240.aspx)

３．２ 誰が為の新技術

ClickOnce と WCF、XML を活用することで、Web アプリからシームレスにクライアントアプリを起動することが可能となる。技術隆替の果てに混在してしまった Web アプリとクライアントアプリが融合するのである。これによりユーザーは Web アプリなのかクライアントアプリなのかを意識することなく、各々の業務だけに注力することとなり、業務効率が向上する。当然、今回のスタートであった通信機器を使用しないノート端末を使用しても、ユーザーの作業効率を向上させ、システムの保守性の低下を防ぐことできる。

すなわち、ClickOnce と WCF、XML を活用することでユーザーとシステムの双方に利益のあるシステムの構築が可能となる。システムを「ユーザーをサポートする」という本来の姿に戻すことが可能となるのである。

今回は既存の配信テクノロジーを排除することができず、クライアントアプリは同テクノロジーを利用することとなった。結局、この部分のシステム側の負荷は変わらないのである。後日判明したのだが、ClickOnce は差分を更新することが可能なので、アプリの内部を細分化して作成すれば、配信量を少なくすることが可能となり、既存配信テクノロジーを排除することも出来たのではないかと思うと残念でならない。

今システムでは、ClickOnce と WCF、XML がユーザー要望を実現したが、ユーザーの要望を実現するための近道は、テクノロジーに関する広い知識と新技術への好奇心、そして技術を繋ぎ合わせるヒラメキがもたらすものではないだろうか。

参考文献

- Microsoft　ClickOnce(http://msdn.microsoft.com/ja-jp/library/142dbbz4.aspx)
- Microsoft　WCF(http://msdn.microsoft.com/ja-jp/library/bb310550.aspx)
- あいおい保険システムズ(http://www.ioi-systems.co.jp/index.html)