

生命保険業界におけるレガシーシステムの課題と クラウドネイティブ化への移行戦略

日本アクチュアリー会 IT 研究会第 4 グループ

<担当委員>

デロイト トーマツ リスクアドバイザリー合同会社	長瀬 正憲
三菱 UFJ 信託銀行株式会社	松岡 靖之

<メンバー>

明治安田生命保険相互会社	本山 智暉
明治安田生命保険相互会社	飯田 信忠
富国生命保険相互会社	新井 るり子
太陽生命保険株式会社	深口 駿
ニッセイ情報テクノロジー株式会社	山田 綾香
ミリマン・インク	星 秀人
株式会社日本ウィルテックソリューション	中里 貴央

目次

I. 生命保険業界におけるレガシーシステムの現状課題
II. レガシーシステム継続による将来リスク
III. クラウドネイティブ化による保険システムの価値向上
IV. 事業目標を踏まえた段階的移行アプローチ
V. 結論
付録：保険会社向けクラウド移行成熟度評価フレームワーク
謝辞
参考文献

I．生命保険業界におけるレガシーシステムの現状課題

デジタル技術の急速な進化と市場環境の変化が進む現代において、生命保険業界は重大な転換点に立っている。長年にわたり業界の基幹業務を支えてきたメインフレームはレガシーシステムと呼ばれるようになり、今や企業の競争力維持と事業継続性の観点から深刻な課題となっている。

1980年代から構築・運用されてきたレガシーシステムは、高度に複雑化した商品設計や、多様化する顧客ニーズへの迅速な対応を困難にし、「2025年の崖」として経済産業省が警鐘を鳴らす状況へと企業を追い込んでいる。

経済産業省の「2025年の崖」レポートで指摘された主要な課題は、以下の3点である。

- 老朽化したシステムの維持管理コストの増大
- レガシーシステムがデジタル化の足枷となることによる競争力低下
- 旧式システムに精通したエンジニアの高齢化と後継者不足

アクチュアリー会法人会員へのアンケート調査結果からも多くの保険会社がレガシーシステムの課題を認識しながら、具体的な移行戦略の策定と実行に苦慮している実態があきらかとなった。本研究では、日本の生命保険業界が直面するシステム刷新の課題とその対応策について、具体的な実態調査と戦略的展望の両面から検討する。

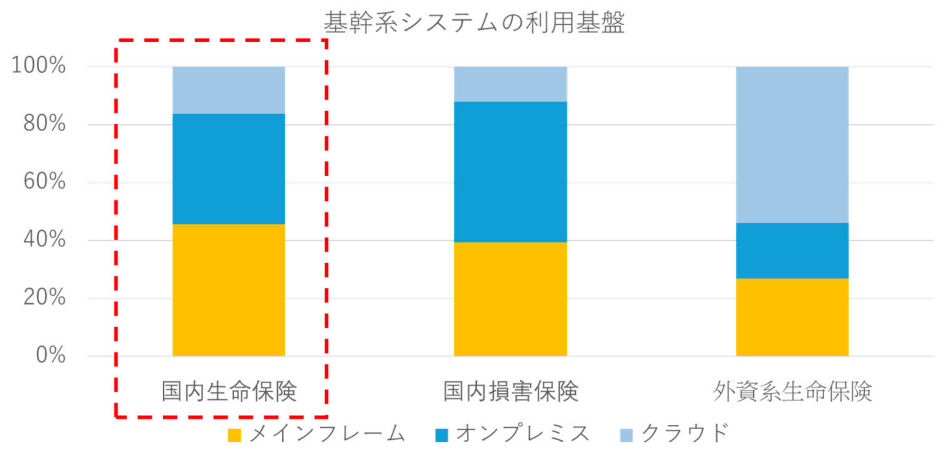
＜本論における各種用語の解説＞

用語	解説
クラウド	サーバーや IT 機器を自社で所有しなくても使えるインターネットサービス。IT ベンダーなどがサーバーやソフト、パソコンの仮想環境などの IT リソースを用意し、インターネットを利用したサービスとして提供する運用方式。（本論文では SaaS、PaaS、IaaS すべてを含む）
オンプレミス	サーバーや IT 機器を自社で所有し運用すること。自社のシステム部からサーバー管理者などを任命し日々の運用をおこなうのが一般的
レガシーシステムとメインフレーム	本論においてレガシーシステムとメインフレームはほぼ同義であるが、メインフレームは狭義のレガシーシステムとして具体的課題や事例を用いるのに使用する。レガシーシステムは広義のメインフレームとして老朽化した大規模基幹システムの抱える課題を論じる際に用いている。
クラウドネイティブ	クラウドネイティブとは、クラウド環境を最大限に活用するために設計されたアプリケーションやサービスの開発・運用アプローチを指す。Cloud Native Computing Foundation（CNCF）によれば、クラウドネイティブ技術は、「スケーラブルなアプリケーションをモダンで動的な環境で構築・実行するためのもの」と定義されている。企業がクラウドネイティブ化を進めることで、迅速なイノベーション、市場投入までの時間短縮、運用コストの最適化などのビジネス価値を創出することが可能となる。
モダナイゼーション	モダナイゼーション（Modernization）とは、旧来の IT システムやアプリケーションを現代の技術標準や要件に適合するよう更新・再構築するプロセスを指す。レガシーシステムの問題点（保守性の低さ、拡張性の制限、セキュリティリスク等）を解消し、ビジネス価値の向上と競争力の維持を図ることを目的とする。モダナイゼーションは単なる技術的な刷新にとどまらず、組織文化やビジネスプロセスの変革を含む包括的な取り組みとなる場合が多い。

1. メインフレームの利用実態

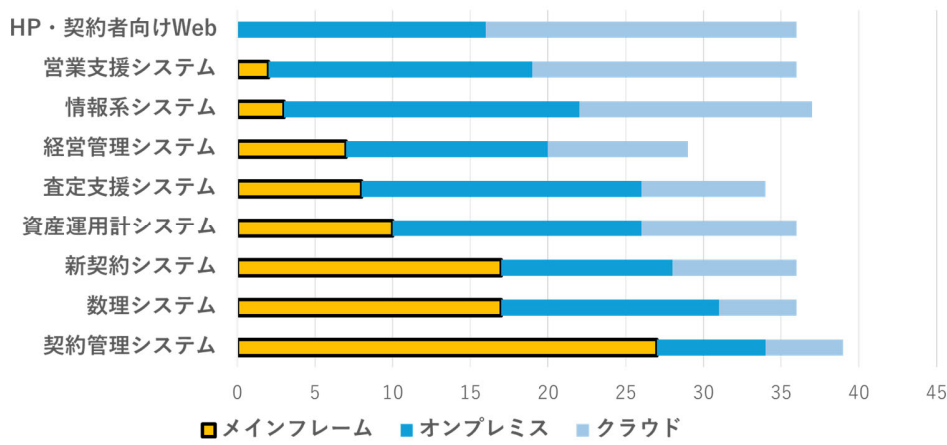
メインフレームの利用実態を把握するため日本アクチュアリー会法人会員の生命保険、損害保険会社 44 社のアンケートを行った。まず、生命保険業界におけるメインフレームの利用状況は図

I -1 の通りとなった。外資系生命保険会社に比べて国内生命保険会社のメインフレームの利用率が高いことが分かる。



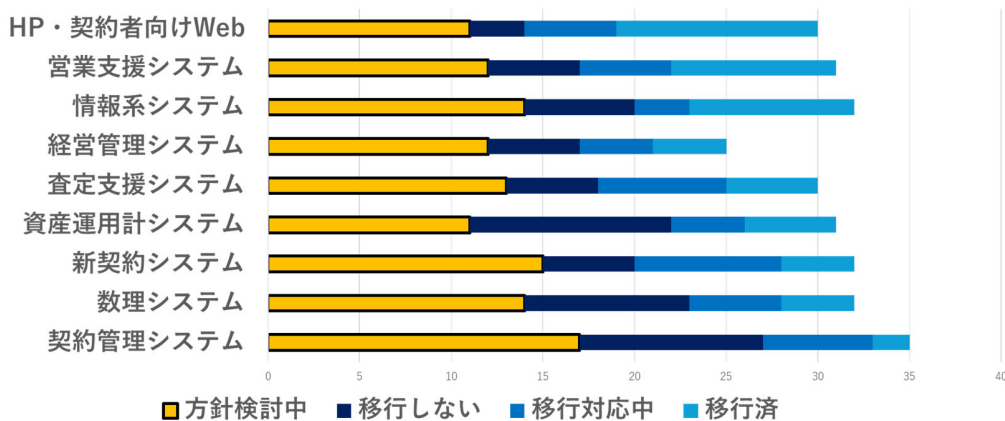
【図 I -1 ：（アンケート結果）基幹系システムの利用基盤】

システム別に集計した図 I -2 では新契約、数理、契約管理系システムのモダナイゼーションが進んでおらず、メインフレームの利用割合が多いことが分かる。一方、新契約、数理、契約管理系システム以外では、クラウドの利用が進んでいる実態が把握できた。



【図 I -2 ：（アンケート結果）システム別の利用基盤】

クラウド移行の検討状況に関する調査結果（図 I -3）では、方針検討中の割合が全体的に大きいことが分かった。



【図 I -3：（アンケート結果）システム別のクラウド移行検討状況】

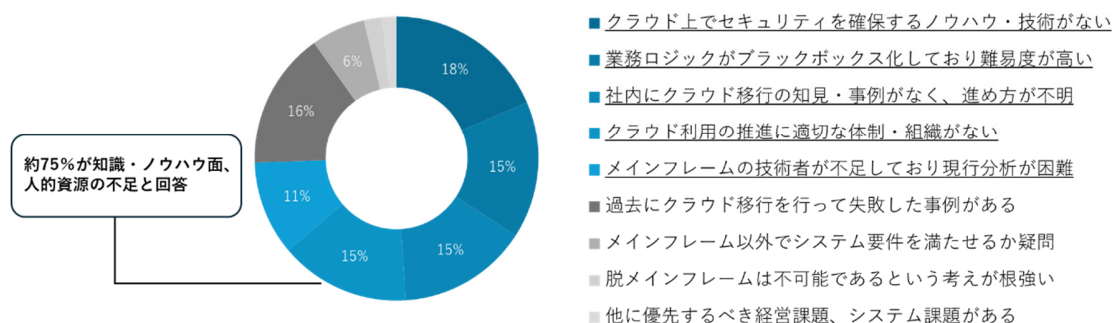
2. 「2025 年の崖」が保険事業に与える影響

前述のアンケート結果の通り、メインフレームのクラウド移行を検討している企業が多いにも関わらず、クラウド移行が思うように進まないのはなぜか。生命保険業界に関わらず、クラウド移行には対応が困難な事情がある。一般的な課題として大きく 4 点を挙げる。

(1) 人材の課題

経済産業省の「2025 年の崖」レポートにおいても、IT 人材の枯渇が叫ばれる中、現業のシステム開発だけで手いっぱいの現場が多くあり、クラウド移行に投下できる人材は不足している。また現行仕様を熟知するベテランも減少しており、クラウド移行を実施する前段階として現行仕様の分析をする人材がいない。さらにクラウド移行には専門知識が必須となるが、クラウド開発を熟知した人材が不在なため、クラウドベンダーの有識者など外部要員に頼らざるを得ないといった現状がある。この現状は、当研究班が実施したアンケート結果にも表れており、クラウド移行検討時の課題・将来的な懸念事項として約 75%が知識・ノウハウ面や人的資源の不足を挙げている。

クラウド移行検討時の課題・将来的な懸念事項（全41社回答、第1位～3位までを選択）



【図 I -4（アンケート結果）クラウド移行検討時の課題・将来的な懸念事項】

(2) 組織の課題

クラウド移行は業務改革を伴うため、事務部門の業務の整理・再構築が必要である。システム部門の一存で移行を進めることはできず、関係部門・関係者は多岐にわたり、プロジェクトは大規模なものになる。さらに業務の整理・再構築を進める中で、社内の合意形成に苦慮する可能性が考えられる。例えば、事務部門の業務内容が大きく変化し、業務自体が変わることによる運営上の負担を意識して、事務部門が業務の整理・再構築に非協力的となることが想定される。

(3) 移行の課題

クラウド移行には、移行対象のシステムや周辺システムの影響調査や、移行に伴う意図しない変更を避けるために大規模な回帰テスト¹が必要となる。また移行対応中は、抜け漏れのない機能移行のために、現行システムの改修に制約が生じる。さらに周辺システムの並行開発を考慮する必要がある。移行中の有事の際のシステム環境を、新環境とは別に並行で用意しなければならないケースもある。

¹ 回帰テスト（Regression Testing）とは、ソフトウェアの修正や機能追加、環境変更などの後に、既存の機能が正常に動作し続けていることを確認するためのテスト手法である。新たな変更によって意図せず既存機能に不具合（リグレッション）が発生していないかを検証する目的で実施される。

(4) クラウドの特性による課題

パブリッククラウドサービス²を利用する場合は、定期的なメンテナンスやサービス障害など、自社のコントロールが及ばない領域で問題が生じる可能性がある。またメインフレームと比較した際に、レイテンシー³などの問題から応答速度が低下するなどの性能劣化の懸念があり、各社のSLA⁴に耐えうるための対応が必要になる。

以上の課題から、クラウド移行は一般的にコスト・リスクが高く、多くの企業においてクラウド移行へ踏み切る経営判断ができない状況が続いている。

3.生命保険業界固有のシステム要件と技術的負債

一般的なクラウド移行の課題に加えて、生命保険業界にはクラウド移行を困難とする特有の課題がある。生命保険業界が所有するシステムには4つの特徴があり、生命保険業界において、システムのモダナイゼーションの難易度を高めている。

(1) 販売チャネルの多様さ・複雑なインターフェイス

生命保険業界においては、保険商品の顧客への訴求力が売上に直結するため、顧客への積極的なアプローチを実現すべく販売チャネルを多様に用意しているケースが多い。営業職員が使用する営業活動支援システム、問い合わせを受け付けるコールセンターシステム、顧客から直接申し込みを受け付けるインターネットシステムなどが例として挙げられる。メインフレームを移行する場合は、この販売チャネルシステムそれぞれのデータ形式、データ連動の方式などのインターフェイスを考慮した上での方針決定が必要である。

² パブリッククラウドサービスとは、サードパーティのクラウドサービスプロバイダーが所有・運営するインフラストラクチャ上で提供される、インターネットを通じて一般に利用可能なコンピューティングリソースやサービスを指す。これらのサービスはマルチテナント方式で運用され、複数の企業や個人が同じ物理リソースを共有する形で利用する。主要なパブリッククラウドプロバイダーには、Amazon Web Services (AWS)、Microsoft Azure、Google Cloud Platform (GCP)、IBM Cloud などがある。

³ レイテンシー (Latency) とは、情報通信システムにおいて、データの送信から受信までの遅延時間、あるいは命令実行から結果が得られるまでの時間のことを指す。

⁴ SLA (Service Level Agreement、サービスレベル契約) とは、ユーザーがサービスを提供する者との間で、サービスの品質について合意する契約のこと。

(2) ブラックボックス化した仕様（複雑な商品性・長期に亘る商品処理の維持）

生命保険システムは各生命保険会社の考案した複雑な商品性を実現するため、例えば保険料計算、責任準備金計算を担う数理システムなど、前提として生命保険知識なしでは理解ができない非常に複雑なロジックを含んでいることがしばしばある。保険種類毎にシステムを分岐させて構築しているケースも多く、仕様のブラックボックス化に陥りやすい。

また生命保険においては、基本的に加入した被保険者が全員脱退（死亡脱退を含む）するまで商品維持する責任を負っているため、各商品処理の維持保守期間が長期化する。保険商品の販売停止後にすぐ当該システムの機能を止めて廃止対応することはできない一方で、新商品開発当初の開発メンバーは定年により引退し、設計書等のドキュメント整備が追い付かないケースが散見される。このことはブラックボックス化を加速させていると考察できる。

(3) 個人情報を含む大量データの長期保管

生命保険システムにおいては、契約期間が長期間となるため数十年に渡る大量の契約データおよび保有期間における全異動データを保管することが必須要件となる。加えて、機微情報を含む顧客データの保管も必要である。このデータ保管には、金融機関としての強固なセキュリティ対策が必要であり、安易な外部サーバーへのデータ移行を阻害する要因となりえる。

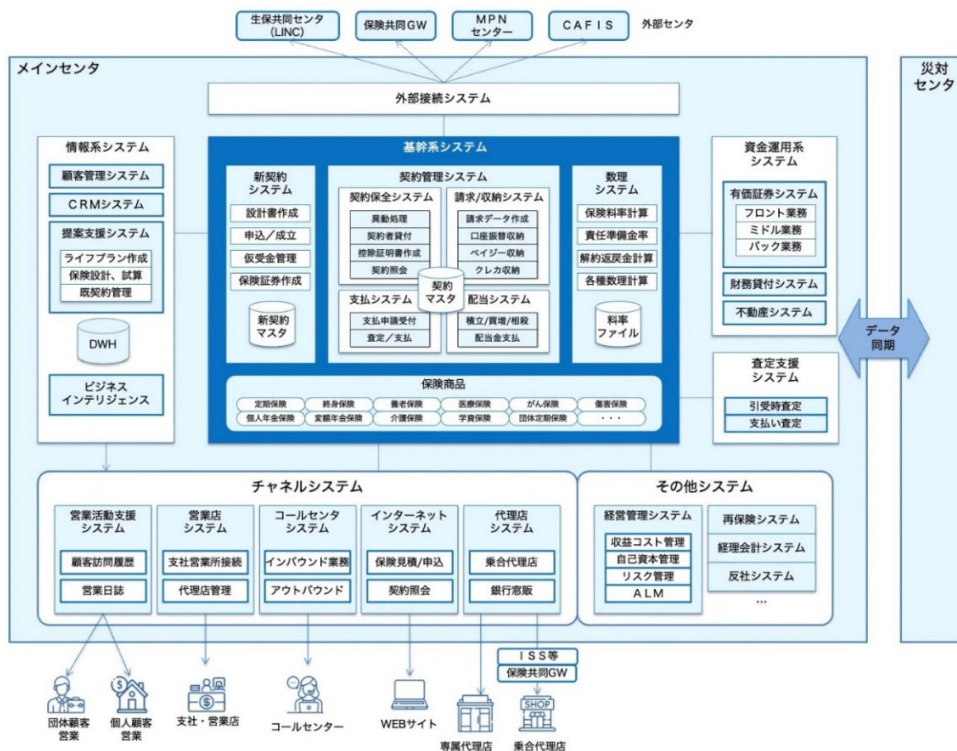
(4) 金融機関として遵守すべき要件（業務継続・法令）

生命保険システムは保険料の収納、保険金の給付など顧客との金銭のやり取りを含む金融システムのため高精度の処理を求められる。人の命に係わる商品を扱っているからこそ、堅牢なBCP対策⁵によっていかなる状況でも業務継続性を維持する必要がある。メインフレームにおいては、複雑なバッチ処理⁶によって、各生命保険会社のシステム要件に合わせたシステム稼働を実現しているが、このような処理をクラウド環境でそのまま実現することが難易度を高めていると考えられる。また各法制度の改正にも速やかに対応する必要がある。移行によって安全性やシステム精度を担保できないことは避けなければならない。

⁵ BCP（Business Continuity Plan）対策。企業が災害や緊急事態に遭遇した際に事業を継続するための計画のこと

⁶ バッチ処理（Batch Processing）とは、データやトランザクションを一定量または一定期間ごとにまとめて一括処理する計算処理方式である。リアルタイム処理とは対照的に、データの収集、処理、出力が連続的ではなく、事前に定義された条件が満たされた時点で実行される。典型的なバッチ処理の例としては、夜間の会計処理、月次請求書生成、データウェアハウスの更新、大規模なレポート作成などが挙げられる。

以上のような生命保険システムの特徴【図 I -5】がクラウド移行、システムのモダナイゼーションをより一層難しいものとしている、という現状課題が確認できた。



【図 I -5：生命保険システム鳥観】

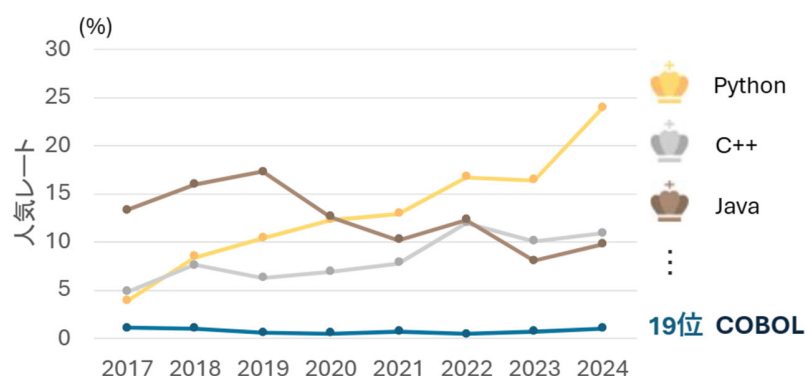
II. レガシーシステム継続による将来リスク

1. COBOL エンジニア不足と知識継承の断絶

前章ではレガシーシステムの現状課題によりシステムのモダナイゼーションおよびクラウド移行が困難な状況を確認してきた。続けて本章ではレガシーシステムをこのまま継続することが現実的に可能であるかについて考察してゆく。まず人材の課題から取り上げていくと、COBOL エンジニア不足の問題が挙げられる。メインフレームを維持する場合、そのシステムを開発・保守運用するシステムエンジニアの確保が不可欠となる。このシステムエンジニア確保の困難さについて、2つの面から考察する。

(1) 新規 COBOL エンジニアの調達困難さ

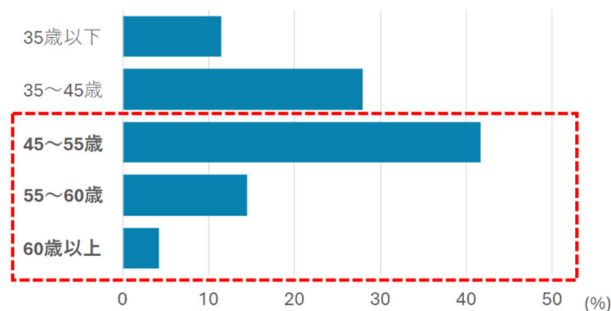
図Ⅱ-1 は、プログラミング言語の人気度を毎月測定する指標を提供する、TIOBE インデックスのデータを参考に作成した、2024 年 12 月時点の人気 TOP3 のプログラミング言語、および COBOL 言語のトレンド推移である。Python や C++、Java などのオープン系言語が TOP3 にランクインしている一方で、COBOL 言語は 19 位と比較的低い順位で横ばいであることが確認できる。このことから、新規に COBOL 言語を学ぼうと考えているシステムエンジニアは少ないことが示唆され、メインフレームを維持するために新たな COBOL エンジニアを確保することは今後難しくなることが考えられる。



【図Ⅱ-1：人気言語 TOP3 と COBOL のトレンド変動(2024 年 12 月)】

(2) 既存 COBOL エンジニアの高齢化

図Ⅱ-2 は、ロイターが 2017 年に調査した、米国の金融業界の COBOL エンジニアの平均年齢を示したグラフである。このグラフから 45 歳以上の層が全体の 60%以上を占めており、COBOL エンジニアの高齢化が進んでいることが分かる。また、本調査が 7 年以上前のデータであることを加味すると、該当層の多くはすでに退職年齢になっていることも考えられ、COBOL エンジニアの不足はより一層進むことが示唆されている。



【図Ⅱ-2 金融業界の COBOL エンジニアの年齢層(2017 年)】

2. 保険システムの技術的負債

次に、ビジネス機会の損失の課題について取り上げてゆきたい。生命保険会社として市場競争力のある新しい商品やサービスを展開するには VUCA⁷時代の顧客が求める保険機能および保障内容に対応するため、新しいサービスや新規技術を導入・利用したい場面が生じると考えられる。その際、自社システムのデータ活用、外部システムへのデータ連携が求められる。レガシーシステムのままこれらの対応をしようとすると、次に挙げるような様々な技術的負債が課題として生じる。

(1) データ

メインフレーム上で長期間保管され更新を繰り返されたデータは、複雑化している傾向が高い。複雑化の要因の一例を挙げるとすれば、メインフレーム上のレガシーシステムで多く使用されている COBOL 言語においては、テーブル上のデータ項目を再定義⁸する手法がしばしば利用されており、1つのデータ項目を複数の名称で再定義しながら使用する、入れ子式のデータ管理を生み出している。

⁷ VUCA（ブーカ）とは、Volatility（変動性・不安定さ）、Uncertainty（不確実性）、Complexity（複雑性）、Ambiguity（曖昧性）の頭文字を取った造語であり、現代のビジネス環境や社会情勢の特徴を表す概念である。VUCA の時代では、過去の経験や従来の分析手法だけでは対応しきれない予測不能な変化に、組織や個人が常に直面することになる。

⁸ REDEFINE(再定義) 句と呼ばれる。項目名を再定義し、別の項目名を定義することが出来る。

例〔HON-HOKENRYO（本人保険料） PIC9（10）.

*R HAI-HOKENRYO（配偶者保険料） PIC 9（10）.

*R KO-HOKENRYO（子ども保険料） PIC 9（10）.〕

特に生命保険業界においては保険商品ごとに管理すべき情報が異なるため、固定長ではなく可変長でデータを管理し、商品毎にデータ項目を再定義することで、レコード単位の使用領域を少なくし、できるだけリソースを節約する努力を重ねている。そのような生命保険システム上には、独自のデータ構成をもち、同じデータが別の名称で重複して点在し、正規化が適切に行われずにまま運用されているデータベースも多くある。

また特に国内企業のシステムにおいては、個々のデータ名称に英語ではなく日本語名称または日本語+英語名称〔例：HIHOKENSHA-BANGO(被保険者番号)、KIGO-SHOKENNO(記号証券番号)〕が使用されているケースも散見される。このようなデータ名称では海外企業・ソフトウェアやシステムエンジニアによるデータ解析は不可能に近く、せっかく長年蓄積した生命保険会社の財産ともなりうるデータは、再整理をせずに利活用することが難しい。

(2) ロジック

現在国内企業で利用されているメインフレーム上のレガシーシステムは、1980年代から1990年代に初期開発されたものが主流で、約40年以上前の古い設計思想がロジック上残存している。また言語としても大半がCOBOL言語を使用していることから、Java言語に代表されるオブジェクト指向⁹とは異なり、構造型プログラミング¹⁰でコード設計されている。このことが処理を冗長的に記述する性質を有しており、レガシーシステムはJava言語などモダン言語からネイティブに設計・開発されたシステムよりも膨大なステップ数となる傾向が高い。

また長期利用される中で、過去のシステム不備に対して行った臨時的なパッチ対応が残存し、プログラム上に点在しているケースも考えられる。当時対応したシステムエンジニアが現場を離脱し、紙ベースの保守ドキュメント管理が不十分だと、システム解析が不能となって抜本的な改修に踏み出せないようなケースは多い。

このようなロジックの複雑性・重複性がシステム開発に向けた現行分析、その後の保守の難易度を高めている。

⁹ 各処理を部品（クラス）として呼び出すプログラミング指向

¹⁰ すべての処理を明示的かつ段階的に記述するプログラミング指向

(3) システム環境

国内大企業については自社のデータセンターを設立し、自社内でメインフレーム・サーバーを含めた運用・保守を行っているケースが多くある。また特に金融業界においては、システムの可用性を高める目的やユーザーから要求される高いセキュリティ対策のために、パブリックな環境ではなく自社専用のシステム基盤を構築しているケースが多い。このようなクローズドなシステム環境では、外部サービスと連携するハードルが高い。

例えばメインフレームとパブリッククラウド上の外部システムを直接接続することは、インターフェースの違いから難しいため、一旦オンプレミスシステムを介在して外部接続するような冗長なシステム構成をとることとなる。この場合、それぞれの接続部分でのインターフェースのすり合わせや影響調査、テストが必要になる。日本企業においては組織や業務を縦割りとする傾向から各システムがサイロ化¹¹しており、部門ごとに管理していることも想定される。接続システムが増えるほどに各担当間での調整事項は増え、開発の難易度が高くなる。

このように、新規サービスや新規技術は、メインフレーム上に実装したシステムレガシーシステムと相性が悪く、社会的ニーズに合致した機動的かつ斬新な保険ビジネスを展開しようとしても、レガシーシステムが足枷となる。

3. 商品開発サイクルの長期化

次に、生命保険システムにおける新商品開発を例に挙げて、保険商品の開発サイクル長期化について検討する。

(1) 顧客のニーズ分析

新商品開発には事前の顧客ニーズ分析を必要とする。しかしレガシーなシステムにおいては、前述のとおりデータ仕様やデータ構成の問題で、本来であれば潤沢な情報元となり得る既契約データに保険業法以外の情報（マーケティングやニーズ分析のための顧客特性情報）を含めることができず、長年蓄積したデータを十分に活用することができないという課題を抱えている。そのため新規にデータの収集や外部情報機関からのデータ買い取りが必要となり、調査や分析費用においてコス

¹¹ サイロ化（Siloing）とは、組織内の部門や機能が独立して動作し、他部門との連携や情報共有が限定的または欠如している状態を指す。

ト増加を招くこととなる。

(2) 新商品開発

レガシーなシステムにおいては、複雑化した大規模なロジックの特性により小規模な開発においても多大な影響調査やテストを要するケースが散見される。既存商品向けのロジックやシステムが密接につながっているため、新商品において同じ計算処理を実装する場合でも既存商品への影響が懸念される。そのため実装には細心の注意を要する上に品質担保には膨大なテストケースが必要である。このことから開発期間は長期間化することが常態化し、市場投入前に開発初期の想定していた市場環境が変動して、時代遅れな新商品をリリースするリスクが高まる。また競合他社によって類似商品が先に販売されてしまうリスクも高くなる。

(3) 営業力強化

新商品の実装が無事に完了できたと仮定して、営業支援システムを追加開発しようとする場合を考える。前述のように新規に外部サービスを利用しようとしても連携が困難であることが想定される。また独自システムを新規構築するとしても、メインフレーム側の制約による障壁が想定される。例えばツールへの最新情報のデータ反映に数日かかるなど営業職員の機動力を損なってしまうケースが想定され、商品販売の機会損失リスクが懸念される。

開発段階	顧客のニーズ分析	新商品開発	営業力強化
想定状況	・長年蓄積したビッグデータの活用困難 ・新規にデータ収集、加工	・小規模開発でも膨大な影響調査・テスト要 ・開発期間が長期化	・外部サービスの導入困難 ・メインフレームの制約で自由な設計不可
想定結果	・初期コスト高騰 ・データ（資産）の価値損失	・市場への商品の適応性低下 ・競合他社の先行発売	・ユーザーのニーズに合わないツールの実現 ・商品販売数低下

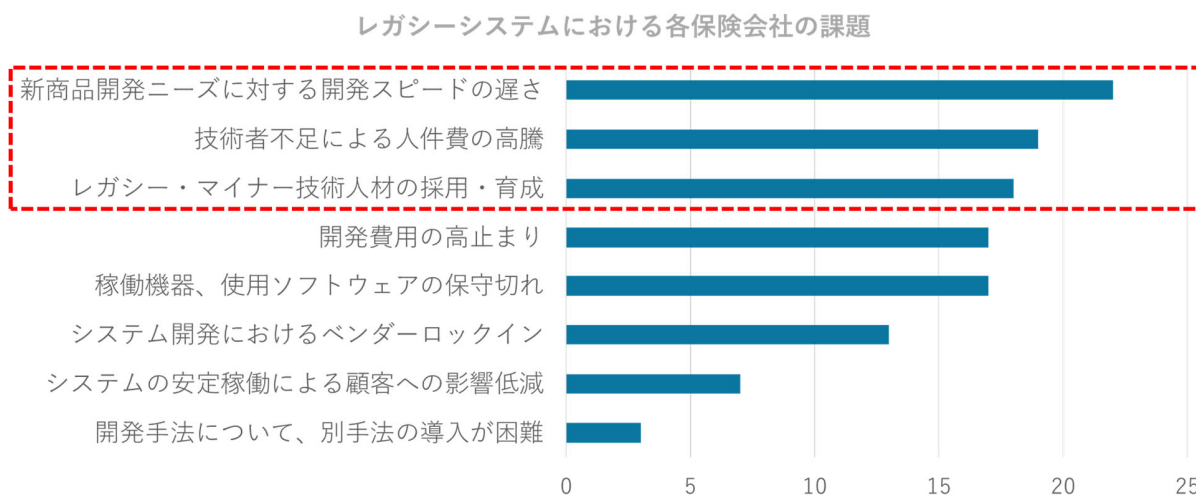
【図Ⅱ-3：生命保険業界の新商品開発におけるビジネス機会の損失例】

以上の検討事例から、レガシーシステムを維持したまま商品開発サイクルの長期化を改善するのはシステムの仕組みとして困難であると考えられる。

4. システム維持コストの高止まりと収益性への圧迫

システムの自社保有（オンプレミス）は、商品開発コストの損失、商品開発期間の長期化、要員調達の困難さ・調達コストの増加、基盤運用コストの増加、基盤調達費の高騰などにより運用・保守コストが高止まりする要因になる。その結果、初期投資も高く、運用保守コストも高い状況が続いてしまう。JUASの2024年企業IT動向調査では、日本企業のIT予算の内、約80%は運用・保守コストとなっている。攻めのIT投資を行いたいとしても原資を確保できず、ビジネス展開において大きな制約を抱えることになっている。

最後にレガシーシステムにおける各保険会社の課題意識についてアンケート結果【図Ⅱ-4】を確認する。一番課題視されている事項として「新商品開発ニーズに対するシステム開発スピードの遅さ」が挙げられている。また、「技術者（システムエンジニア）不足による人件費高騰、採用・育成」も上位に挙がっており、今までに挙げた課題が各保険会社においても意識されていることが確認された。



【図Ⅱ-4：（アンケート結果）レガシーシステムにおける各保険会社の課題】

Ⅲ. クラウドネイティブ化による保険システムの価値向上

1. 商品開発・提供サイクルの迅速化と顧客ニーズへの対応力強化

前章で確認した通り生命保険業界においては、商品の設計・開発・提供までに長い期間を要することが一般的であった。しかし、市場環境の変化や顧客ニーズの多様化に対応するためには、これまでのような長期的な開発サイクルでは競争力を維持することが難しくなってゆくことが想定される。本章からクラウドネイティブ技術を活用することで、保険商品の開発スピードを向上させ、より柔軟な提供が可能となる点を示してゆく。

（１）クラウドネイティブ化による商品開発の迅速化

クラウドネイティブアーキテクチャ¹²では、従来のモノリシック¹³な開発手法から脱却し、マイクロサービス¹⁴を活用した柔軟なシステム開発が可能となる。これにより、以下のようなメリットが得られる。

- **開発の並行実施が可能:** サービスを細分化し、個別の開発チームが独立して作業できるため、開発期間の短縮が実現。
- **CI/CD による迅速なデプロイ:** 継続的インテグレーション・デリバリー（CI/CD）¹⁵の活用により、新機能や改善を迅速に市場投入。

¹² アーキテクチャ（Architecture）IT 分野では情報システムの基本設計や構成要素の構造を表す概念として広く用いられる。

¹³ モノリシック（Monolithic）とは、「一枚岩」や「単一体」を意味する言葉で、ソフトウェアアーキテクチャの文脈では、アプリケーションの全機能が単一のプログラム単位として設計・実装・デプロイされる構造を指す。モノリシックアーキテクチャでは、ユーザーインターフェース、ビジネスロジック、データアクセス層などのコンポーネントが密結合しており、一体として動作する。従来型のエンタープライズアプリケーションの多くはこの構造で開発されてきた歴史がある。

¹⁴ マイクロサービス（Microservices）とは、アプリケーションを小さく独立した複数のサービスに分割する分散型ソフトウェアアーキテクチャのアプローチである。

¹⁵ 継続的インテグレーション・デリバリー（CI/CD）とは、ソフトウェア開発における実践的なアプローチであり、開発プロセスの自動化と頻繁な統合・デリバリーを通じて品質向上とリリースサイクルの短縮を実現する方法論である。CI（継続的インテグレーション）は開発者による変更を自動的に検証し、共有リポジトリへ定期的に統合するプラクティスであり、CD（継続的デリバリー/継続的デプロイメント）はテスト済みのコードを自動的に本番環境またはその手前の環境へ配布するプロセスを指す。両者を組み合わせた CI/CD パイプラインは、コード変更のコミットから本番環境へのデプロイまでを一連の自動化されたステップとして実現する。

- **コンテナ¹⁶技術の活用:** アプリケーションをコンテナ化することで、環境依存の影響を排除し、開発スピードを向上。

(2) 顧客ニーズへの迅速な対応

デジタル化の進展により、顧客の期待値は高まり、多様なサービスをリアルタイムで提供する必要がある。クラウドネイティブ技術を活用することで、以下のような対応が可能となる。

- **短期・小規模開発の繰り返しが可能:** 小規模な単位で開発を行い、事業環境の変化を即座に反映しながら段階的に改良を進めることが可能。
- **データドリブンな商品設計:** クラウド環境でビッグデータや AI を活用し、契約者の行動分析や市場動向を即座に反映した商品設計が可能。
- **パーソナライズドサービスの提供:** API を活用した外部サービスとの連携により、個別顧客のニーズに応じた商品やプランの提供が容易。
- **オムニチャネル対応の強化:** Web、モバイル、コールセンターなど、あらゆるチャネルを統合し、一貫した顧客体験を提供。

2. システム基盤コスト構造の最適化と投資効率の向上

少子化社会における持続可能性の向上も欠かすことのできない論点である。保険業界においても IT 人材の高齢化や採用難が進む中、システム基盤のコスト構造を最適化し限られた人材で効率的な運用を実現することが求められている。クラウドネイティブ化の主要な利点の一つが自動化・省力化であり、従来と同等以上のサービス品質を、より少ない人員で維持することが可能となる。

具体的には、DevOps の導入により、開発から運用までの一貫した自動化が実現し、人的作業を最小限に抑えることが可能となる。また、クラウドネイティブアーキテクチャによる柔軟な資源配分により、システム運用の効率化が図れることが挙げられる。これらの取り組みはコスト削減だけ

¹⁶ コンテナとは、アプリケーションとその実行に必要な依存関係（ライブラリ、バイナリ、設定ファイルなど）をパッケージ化した軽量の実行環境である。従来の仮想マシン（VM）と比較して、OS カーネルの共有によりオーバーヘッドが少なく、起動が高速で、リソース効率が高いという特徴を持つ。

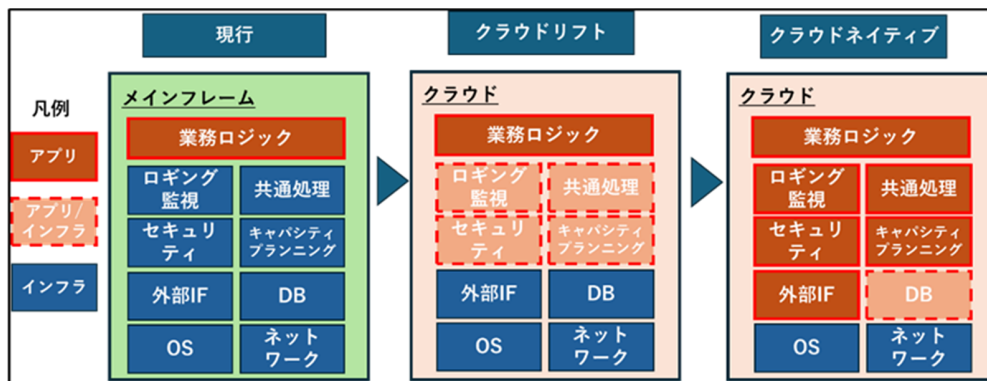
でなく、貴重な人的リソースをより価値創造的な業務に再配分することを可能とし、結果として投資効率の向上にも寄与するものである。

3. デジタル人材の確保・育成と組織活性化

クラウドネイティブ化によって、デジタル人材の確保・育成と組織活性化の面においてもメリットが得られる。レガシーシステムのクラウドネイティブ化を進めていくにつれ、採用する技術や考え方には大きな変化が生じてくる。その過程が、業務刷新・改革に必要なデジタル人材の獲得につながり、事業環境の変化に即応できるような組織の活性化を図ることができる。以下に主な変化とその影響について述べる。

(1) 担当・責任範囲の変化

従来のメインフレーム上のモノリシックなアプリケーションにおいては、アプリケーションエンジニアは業務ロジックのみを、インフラエンジニアはシステム基盤部分のみを担当すればシステムの開発運用保守は事足りた。しかしクラウドリフト¹⁷、クラウドネイティブ化が進むにつれて、業務ロジックのみ留意していたアプリケーションエンジニアが、責任を持つべき領域は次第に拡大する。下記に担当領域の変化を図にして整理した。



【図Ⅲ-1：クラウドネイティブ化に伴う担当領域の変化イメージ】

¹⁷ クラウドリフト（Cloud Lift）とは、既存のアプリケーションやITインフラストラクチャをオンプレミス環境からクラウド環境へ移行する戦略的アプローチを指す。「リフト&シフト」（Lift and Shift）とも呼ばれ、アプリケーションの基本的なアーキテクチャや構造を大きく変更することなく、そのままクラウドへ移行することを特徴とする。次章にて追加で解説する。

セキュリティを例に挙げる。メインフレーム上に実装されたシステムにおいては、アクセス制御や外部との通信については、インフラエンジニアの領分であり、アプリケーションエンジニアが留意すべき領域は少ない。一方で、クラウドに移行し、システムをオープン化すると、外部からの攻撃に備えた実装を検討する必要が生じる。更にクラウドネイティブ化すると、サービスがマイクロサービス化し API¹⁸でサービスを提供するようになるためネットワーク面でのセキュリティの考慮が求められるようになる。

続けてキャパシティプランニングについて考察する。従来であればインフラ側で事前に用意された CPU やメモリ・ストレージを前提にアプリケーションを実装すれば十分であった。一方で、クラウド環境では、計算リソースは柔軟に確保できるため、業務の観点から見たリソースの事前設計・検討が必要となる。業務処理量が通常どのくらいあるのか、処理のピークは発生しうるのか、あるいは予測不可能であるか、そういった観点はインフラエンジニアではなく、アプリケーションエンジニアの領分である。クラウドリフトをした直後の状態においては、アプリケーションは依然としてモノリシック（もしくはそれに近い状態）であると想定され、その時点ではインフラエンジニアの担う領域が大きい。クラウドネイティブ化が進み、サービスの分割が進むと、アプリケーションエンジニア側主体で、キャパシティを検討・チューニングする必要が次第に生じる。

完全にインフラから切り離してアプリケーションの領域に変換されるべきか否かは、対象のシステムの特徴や、そのシステムを保有する企業の組織体制に依存するところがあり、一概に言えることではない。しかし、担当領域に変化が生じ、徐々にアプリケーションエンジニアが留意すべきことが増えてくる、という点に違いはないであろう。このような保険システムのモダナイゼーション過程が、次世代のシステムに対応できる人材育成につながる。

(2) 開発思想の変化

クラウドリフト・クラウドネイティブ化は、担当領域の変化のみならず、システムを開発・保守・運用する思想にも変化を与える。

¹⁸ API (Application Programming Interface: アプリケーション・プログラミング・インターフェース) とは、ソフトウェアコンポーネント間の通信を可能にする仕様や規約の集合体である。API は異なるプログラム、システム、サービス間の相互作用方法を定義し、開発者がアプリケーションの特定の機能やデータにアクセスするための標準化された手段を提供する。

a. 計算リソースは従量課金される

クラウドにおいて計算資源は柔軟に確保することができるが、一方でそれらの計算資源は従量課金される。オンプレミスでシステムリソースを利用する場合は、その性能の範疇でコスト面を気にせずリソースを利用できたが、クラウドにおいてはリソースの使用量がコストにそのまま反映される。予想外のコスト増を防ぐためにも、無駄なリソース消費が無いが、開発・保守・運用の観点から常に気を配る必要がある。

b. 言語特性の違い

COBOL から Java に変換する場合、両言語は言語特性に違いがある。一方は手続型言語¹⁹、他方はオブジェクト指向型言語²⁰である。実装の前提となる考え方が異なるため、継続的リファクタリング²¹を進めていく上では、留意が必要となる。開発規約やコーディング規約を再整理し、一貫した方針のもと開発・運用・保守を進める必要がある。

c. 開発プロセス・運用プロセスの違い

従来のレガシーシステムでは、ウォーターフォール型開発²²が採用されることが多かった。一方で、クラウドネイティブなシステムの開発においては、アジャイル型開発²³が基本となる。システムの実態によってはアジャイル型開発が適さない場合も存在するが、事業環境の変化に迅速に対応するには、アジャイル型開発が望ましい。

またシステム構成の面においても大きな違いがある。クラウドの特性の一つである、柔軟な計算資源の確保は、クラウドネイティブに対応するならば、IaC²⁷やコンテナを活用して自動的にスケールアウト・スケールイン（コンピュータリソースの量的追加・削減）することとなる。この方法

¹⁹ 手続き型言語（Procedural Language）とは、プログラミングパラダイムの一つであり、プログラムを一連の手順や命令の集合として表現するアプローチを採用したプログラミング言語を指す。

²⁰ オブジェクト指向型言語では、データとそれを操作するメソッド（振る舞い）をカプセル化した「オブジェクト」を基本単位とし、複雑なシステムをオブジェクト間の相互作用として表現する。

²¹ 継続的リファクタリング（Continuous Refactoring）とは、ソフトウェア開発において、コードの外部的な振る舞いを変えずに内部構造を改善する作業を、開発サイクル全体を通じて定期的かつ継続的に行うプラクティスである

²² ウォーターフォール型開発（Waterfall Model）とは、ソフトウェア開発やシステム開発において用いられる伝統的な線形的開発手法であり、各開発フェーズを滝（ウォーターフォール）のように上から下へと順次進行させるアプローチである。

²³ アジャイル型開発（Agile Development）とは、ソフトウェア開発における反復的かつ漸進的なアプローチであり、変化に対する柔軟な対応と顧客価値の早期かつ継続的な提供を重視する開発手法である。

を採用する場合、事前にコンテナイメージを作成する、インフラ構成をコードにする、スケールアウト・スケールイン時の接続維持のためにステートレスな通信²⁸を前提にアプリケーションを構築するなどが必要になる。前提として、こういった知識・技術の獲得が必要となる。

(3) ベテランエンジニアの役割の変化

ここまでアプリケーションエンジニアとインフラエンジニアの担当領域の変化と、前提となる開発思想の変化について述べた。こうした前提の変化にあたっては、担当するエンジニアの再学習が必要になる。組織の観点では、エンジニアの再学習を後押しするような施策が必要である。クラウド知識の習得の推奨や、クラウドネイティブ化を前提としたプロジェクト全体のルール・規約作りが求められる。

新しい技術や思想の導入にあたり、それらの知識を保有するエンジニアの外部採用も十分に選択肢に入るが、一方でクラウドリフト・クラウドネイティブ化にあたっては、既存のシステムの仕様を熟知したベテランエンジニアの参画も必要不可欠である。クラウドリフトにおいては、自動変換できない箇所のコードの解析や仕様面から見たバグの解析、クラウドネイティブ化においては、適切なアプリ分割、モジュール化を進める上でその知見を最大限活用することができる。

以上、クラウドネイティブ化により、エンジニアにおける、担当・責任範囲の変化、開発思想の変化、ベテランエンジニアの再学習について述べた。これらの変化に適切に対処することが、クラウドネイティブ化には求められる。一方で、その過程が、次世代のシステムやビジネスを担うデジタル人材の確保・育成や IT を真の意味で活用した組織活性化につながる事となる。

IV. 事業目標を踏まえた段階的移行アプローチ

前章では保険システムの価値向上のためにクラウドの活用は有力な選択肢となることを示した。一方で、レガシーシステムのクラウド移行は、一般的に高難易度・高コスト・高リスクである。特に生命保険業界においては、複雑なインターフェイス・バッチ処理、ブラックボックス化した業務、超長期のデータ保全などの固有課題が難易度を高めているが本章では生命保険業界におけるクラウド移行の新たなアプローチを提案してゆく。

1.事業目標と移行目的の整合性確保

高難易度な生命保険会社におけるクラウド移行は、まず移行目的を明確にすることが重要となる。クラウド移行自体も長期に渡るプロジェクトとなるが、システムのモダナイゼーションのためには、クラウド移行が完了したのちにも継続的な改善が欠かせないため、短期的なメリットを追求することは適切でないと考えられる。短期的なメリットが得られず、かつ移行目的を明確にしない場合、移行の軸が定まらず、移行に伴う部門間の調整や一貫性を持った移行計画の策定や遂行が困難になることが想定される。結果、クラウド移行を完遂してもクラウドプラットフォームの利点を十分に活用できず、クラウドによって得られるメリットが限定されるばかりでなく、かえって運用コストの増大を招く結果となりかねない。

移行目的の設定は、システムのモダナイゼーションにとどまらず業務の改善、デジタルトランスフォーメーションにも直結する内容となる。つまり、経営戦略・事業目標に直接関係する。その観点から、クラウド移行においては、経営層の強固なコミットが求められる。単純にメインフレームを廃止することを目的とするのではなく、クラウドへと移行することによってどのような自社の課題を解決するのか、どのような事業を展開して社会に付加価値を提供するのか、自社が使える経営資源は何か等を踏まえて、移行目的を明確に定め、全社的に展開・周知する必要がある。

計画実施の段階においても、クラウド移行を進めるプロジェクトが、社内調整をスムーズに進めるためにも体制や組織、人員、資源の面でサポートを行う必要がある。また、クラウド環境への移行が完了し、クラウドネイティブ化を進めるにあたって、継続的な改善活動の推進を可能とするような体制を構築することは欠かせない。なお、具体的にどのような観点で検討が必要か、自組織がクラウドネイティブ化を進めるにあたってどの程度備えられているかを判断・確認するために、「保険会社向けクラウド移行成熟度評価フレームワーク」を作成し文末に付録した。クラウドネイティブ化を進めるにあたり自社の状況を分析し、次ステップとして何を目指すかの参考としてお役立て頂ければ幸いである。

また、移行目的によって、どの程度クラウドを利活用するべきかは変化する。事業目標によっては必ずしもクラウド移行が適切でない場合がある。各目的に応じて、どのような移行方針をとるべ

きかは、下表に整理した。

移行方針	目的
メインフレームを維持	システム改修が殆ど発生しない・新規サービスを展開する予定がない。 もしくは、メインフレーム上のデータのみ活用する。
オープン化	COBOL 人材課題を解決し、人材調達の選択肢を拡大する。 既存システムのメンテナンスに関わる将来的な懸念を解決する。
クラウドリフト	システムリソースの柔軟な増減、ならびにミドルウェアのパッチ適応に伴うシステム基盤の運用・保守コストを削減する。有事におけるシステムの可用性を高める。
クラウドネイティブ化	クラウドの利点を最大限活用し、開発運用保守コストの最適化を図る。新商品や新規サービスの迅速なビジネス展開や DX を推進する。

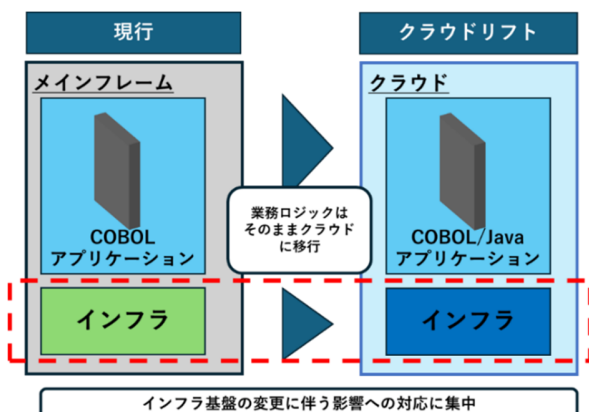
【図IV-1：クラウド移行方針と移行目的】

2. 保険会社の事業特性に応じた最適移行戦略

次に、段階的移行について触れる。高難易度・高コスト・高リスクである生命保険業界のクラウド移行においては、集中して対応すべき課題を切り分けて、リスク分散を図ることが求められる。移行目的や経営戦略に基づき、適切な進め方は各社ごとに検討が必要となるが、一番難易度が高いことが想定される、大企業・大規模システムの基幹システムのクラウドネイティブ化を目指す場合、大きく2つのフェーズに分けて段階的に進める必要がある。

(1) 第1フェーズ：メインフレーム上のシステムのクラウドリフト

段階的移行の第1フェーズとして、メインフレーム上のシステムをクラウド環境に移行する。ここでは、業務ロジックの改修を極力抑えつつ、環境の移行に伴う改変に集中する。具体的には、外部システムとのインターフェイス部分や、セキュリティ、データベースや OS といったシステム構成の変更に伴う改修に注力する。いわゆるクラウドリフトを進める。



【図IV-2：クラウドリフト移行イメージ】

ブラックボックス化した仕様の問題や手続き型プログラミングに特化した冗長なシステム構造の問題は残存するが、それらを一挙に解決することは難しい。まずはインフラ面での改修に集中することで、困難を切り分け、クラウド移行に係るコストと難易度を抑える。

クラウド移行にあたっては、ツールの利用が不可欠である。なぜならば、クラウド移行においては、ソースコードの変換や移行など、業務要件が直接に関与しない領域が多く、いかに作業を自動化・効率化するかが、成功の可否に大いに影響するからである。また、大規模な回帰テストを実施して現行機能保証を図るためにも、テストの効率化は欠かせない。下記に挙げるようなツールを移行計画策定時に検討し、活用していくことが求められる。

概要		例
コード解析ツール	ソースコードの解析を行い、最適なマイグレーション方式を選択する下準備を行う。	・ Micro Focus Enterprise Analyzer ・ AWS Blu Insights
コード変換ツール	クラウドに移行の際、ソースコードの変換を行う。	・ AWS Blu Age
テスト支援ツール	最も工数のかかるテスト工程を短縮するために、テストデータの生成などを行う。	・ Delphix
リプラットフォームツール	メインフレーム上で稼働しているCOBOLソースをそのままクラウドに移行し利用する。	・ Micro Focus Enterprise Server

【図IV-3：クラウドで活用すべきツール】

なお、ツールの利用による作業の自動化・効率化はほぼ必須ではあるが、全てを自動化することは現実的ではない。自動化の困難な部分に関しては、現行仕様の理解に基づいた手作業での改修が発生しうる。その際は、既存のシステム仕様を熟知したベテランエンジニアの活用が強く求められる。

コード変換ツールについて補足する。コード変換においては、言語変換方式も検討が必要となる。大きく分けて、Java に変換する方式、COBOL ライクな文法の Java に変換する方式（Jobol 方式）、移行先でも COBOL を利用する方式のそれぞれがある。メリット・デメリットを下記に整理した。いずれを選ぶかにおいても、移行目的や事業目標に基づく判断が各社において必要になる。

	メリット	デメリット
Java 変換	COBOL 人材問題を早期に脱却できる。	既存人材の活用には工夫が必要
Java 変換 (COBOL ライクな文法)	オープン化しつつ、既存人材を一定程度活用できる。	かえって読みにくい実装になる懸念がある。
移行先でも COBOL を利用	既存の COBOL 人材を活用できる。 移行の速さを重視できる	COBOL 人材問題は解決しない。

【図IV-4：言語変換方式ごとのメリット・デメリット】

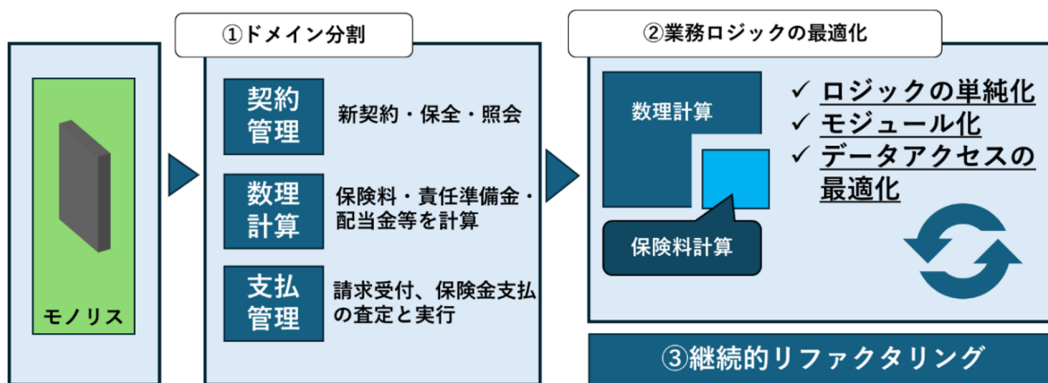
(2) 第2フェーズ：クラウドネイティブ化と継続的リファクタリング

第1フェーズのクラウドリフトが完了し、その後の運用も安定した段階で、第2フェーズを検討する。クラウドリフトの段階では、システム基盤がクラウド環境に単純に移管した状態であり、クラウドのメリットを最大限に発揮している状況とは言えない。この状況では、依然としてレガシーなシステムからは脱却しきれておらず、柔軟な事業展開の観点では制約が大きい。ここから、クラウドネイティブ化を図っていくことになる。

クラウドネイティブなアプリケーションにおいては、サービスを細かく分割するマイクロサービス化を検討していく。ただし、大きなシステムであればあるほど、その作業は困難を極める。また、生命保険業界における最適なサービスの粒度については一般解が無く、模索が必要となる。生命保険業界におけるレガシーシステムのクラウドネイティブ化は先行事例が少なく、また各社のレ

ガシーシステムはフルスクラッチ²⁴されている点からも試行錯誤の域を出ないが、生命保険システムの実例に即して、一つの例を検討した。

クラウドリフトしたモノリシックなシステムを、独立して稼働することのできる大まかなドメインに分割する。例えば、新契約や契約管理、契約内容照会等を一つにして契約管理ドメインに、保険料、責任準備金、配当金等を最適な計算リソースを使用して処理する数理計算ドメインに、請求受付、保険金支払の査定と実行を監督する支払管理ドメインに分割する。



【図IV-5：クラウドネイティブ化と継続的リファクタリングイメージ】

こうして大まかなドメインに分割した後は、各ドメイン内においてリファクタリングを行う。改修頻度の高いロジックからモジュール化、アプリケーションの最適化を進め、徐々にサービス分割を進めていく。以上のプロセスを継続的に行い、リファクタリングを繰り返してゆくことで、最終的にクラウドネイティブ化を目指していく。

この開発プロセスは広義のウォーターフォール開発からアジャイル開発への変更も表している。次世代のモダンなシステムにおいては、システムの最終系が計画段階で示されるのではなく、その時々々の事業環境に応じた最適な姿に有機的に作り替えていくことが求められる。事業環境の変化だ

²⁴ フルスクラッチ（Full Scratch）とは、既存のフレームワークやライブラリ、テンプレートなどを使用せず、完全に一から独自のコードで開発することを指す。「スクラッチ」（scratch）は「ゼロから」という意味を持ち、フルスクラッチ開発では、システムやアプリケーションの全てのコンポーネントを開発者自身が設計・実装する。この開発アプローチは、特定の要件や環境に合わせて最適化された独自のシステム構築を可能にする。

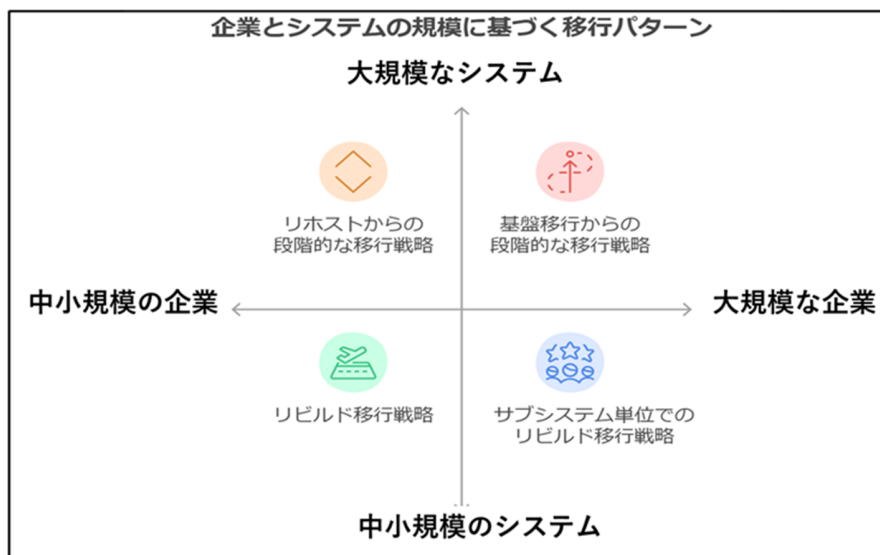
けでなく、クラウドプラットフォームの進化によっても都度最適な状態を目指すことがクラウドネイティブ化の思想的転換と考えられる。

(3) 企業規模別システム規模別の移行パターン

ここまで大規模システムにおいてクラウドネイティブ化を進めることを前提として述べてきた。しかし現実には企業の規模やシステムの複雑性に応じて最適な移行戦略は異なることが想定される。このため本論では、企業環境を考慮した実践的な移行パターンを提案したい。

移行戦略を企業規模とシステム規模という二つの軸に分類することで、それぞれの状況に適した移行パターンが作成できる。図IV-6はこの二軸によって分類された4つの移行パターンを示している。

大企業においては、人的リソースや資金等の経営資源が比較的豊富であるため、長期的な視点での段階的移行が可能である。一方、中小企業では限られた経営資源の中で効率的な移行を実現する必要があると考えられる。またシステム規模の観点では、大規模システムは複雑性が高く影響範囲が広いためリスク分散が必要となる。中小規模システムは影響範囲が一定程度限定的であり、より迅速なアプローチが可能となることからリビルド²⁵による複雑性の解消が期待される。



【図IV-6：企業とシステムの規模に基づく移行パターン】

²⁵ リビルド (Rebuild) とは、システムやアプリケーション機能要件を維持しながら技術的に完全に作り直すアプローチを指す。レガシーシステムのモダナイゼーション戦略の一つとして位置付けられ、「リホスト（環境移行）」「リプラットフォーム（基盤変更）」「リファクタリング（内部構造改善）」など他の手法と比較して、最も抜本的な再構築を伴う。

①大規模システム・大企業 基盤移行からの段階的な移行戦略

COBOL のままクラウド環境にシステムを移行。その後クラウドネイティブ化を進める。超長期間のプロジェクトとなるが企業体力的に可能。リスク分散と有識者の活用に重きを置く。

②大規模システム・中小企業 リホストからの段階的な移行戦略

Java に変換しつつクラウド環境にシステムを移行。その後クラウドネイティブ化を推進する。人材調達面での課題解決に重きを置き、オープン化も併せて行う。

③中小規模システム・大企業 サブシステム単位でのリビルド戦略

業務ロジックの最適化も併せて、サブシステムごとにクラウド移行を行う。
リスク分散を図りつつ、早期のクラウドネイティブ化を目指す。

④中小規模システム・中小企業 リビルド移行戦略

業務ロジックの最適化も併せて、クラウド移行を行う。
リスク分散を図りつつ、迅速なクラウドネイティブ化を目指す。

上記で提案した企業規模とシステム規模による移行パターン分類は一つの目安であるが、実際のクラウド移行戦略を検討する際は移行体制の準備状況や移行計画の作成状況も重要な判断材料となる。自組織の現状を正確に評価、診断するため、本論文では付録として「保険会社向けクラウド移行成熟度評価フレームワーク」を作成した。このフレームワークを活用することで、自組織のクラウド移行に関する準備状況を客観的に評価し、次に取り組むべき重点領域を特定することが可能となる。移行戦略を具体的に推進してゆく際は、ぜひ付録のフレームワークも併せて参照されたい。

V. 結論

1. 本研究の主要な発見と知見の総括

レガシーシステムを保持し続けることには、人材確保の面、柔軟なサービス展開、運用保守コストの高止まりの観点で大きな課題がある。これらの解決のためにも、システムのモダナイゼーション、その前提となるクラウド環境への移行は必要になる。一方で、クラウド移行は一般的に難易度

が高い。とりわけ生命保険業界においては、業界固有のシステムの特徴（複雑なインターフェース・バッチ処理、ブラックボックス化した業務、超長期のデータ保全）があり、その難易度を更に高めている。こうした背景が、クラウド移行の判断や移行計画の策定が難航する要因となっている。

しかし、企業の競争力を高め、持続可能な事業を展開するためには、システムのモダナイゼーションが求められており、その推進のためにはクラウドの利用が必要不可欠である。特に商品開発・新規ビジネスの展開の迅速化を目的とするならば、クラウドネイティブを推進する必要がある。ただし、クラウド移行に唯一絶対の方式は存在しない。各社ごとの経営戦略や経営資源、置かれた事業環境などの前提によって、無数の選択肢が存在する。そのような困難な状況下において成功させるための要素としては、移行目的の明確化と経営層のコミット、段階的移行と継続的リファクタリングの2要素がある、と当研究班では結論づけた。

2. デジタル時代における保険事業の将来展望と課題

最後に、今後の展望について触れる。少子高齢化社会の進展により、今までシステムの開発・保守・運用を担ってきた人材は退職年齢となる一方で、人口の減少により若い人材の確保はより一層困難になることが想定される。少ない人材の活用のためにも、業務の生産性・効率性の向上はより一層求められる。このような観点からも、システムのモダナイゼーションとクラウドの活用は急務である。

一方で、前述の社会的背景からクラウド移行のニーズは一層高まることが想定される。市場規模の拡大を受けて、クラウドベンダーがモダナイゼーション向けのソリューションを強化する可能性がある。また、生成 AI など最新技術や最新サービスも同様に動向を注視するべきである。今は高コスト・高リスクなクラウド移行であっても、クラウドベンダーのサービス拡張や、技術革新でブレイクスルーが起きる可能性がある。継続的に動向を注視しつつ、移行判断や移行計画への取り込みを図るべきである。

また、現状見えている課題への対策や、競争力の確保といった観点だけでなく、現時点では見えていないリスクに備えるという点でも、クラウドネイティブ化は求められる。今時点では予測でき

ない科学技術の進展や社会情勢、ビジネストrendへ備える点でも、柔軟かつ迅速に対応することができるシステム構築は必要不可欠である。

生命保険業界におけるメインフレーム上に構築されたレガシーなシステムのモダナイゼーションは先例が少なく、困難な課題である。既に 2018 年に提示された 2025 年の崖問題が指摘する年は到来しているものの、依然として対応状況は芳しくない。経営層も、業務部門も、システム開発部門も、運用部門も一丸となって取り組まなければならない課題である。しかし、その困難を乗り越えた先に、各個人においても、各社においても、生命保険業界においても、希望の将来がある。社会ニーズに即した新商品・サービスを迅速に展開し、生命保険業界がより社会に貢献できる未来があるのではないだろうか。本研究がその未来の実現のささやかな一助となれば幸いである。

付録：保険会社向けクラウド移行成熟度評価フレームワーク

	評価項目	レベル1 初期段階	レベル2 認識段階	レベル3 計画段階	レベル4 実行段階	レベル5 最適化段階	自社 評価
組織の準備状況	経営層のコミットメント	クラウド移行はIT部門の問題と認識	クラウド移行の必要性は認識されているが具体的計画なし	クラウド戦略が経営計画に組み込み済	移行のための十分な資源が配分済	経営トップが戦略的優先事項として推進	
	組織体制	従来型のシステム部門体制	クラウド検討チームが発足(兼務体制)	クラウド移行の専任組織が設置済	事業部門との協働体制が確立済	組織全体がデジタル変革に適応	
	人材育成	クラウド人材の必要性は認識されていない	外部研修への参加が始まった	体系的な教育プログラムが整備済	人材育成と採用が計画的に進行中	クラウド技術を活用できる人材が十分確保済	
業務プロセスとデータ管理	業務プロセスの最適化	現行業務プロセスをそのまま移行する計画	一部の業務プロセスの問題点が特定済	主要業務プロセスの再設計計画が策定済	業務プロセスのデジタル化が進行中	全ての業務プロセスがデジタルを前提に最適化済	
	商品開発プロセス	システム対応が商品開発のボトルネック	商品開発とシステム対応の問題を認識	商品開発の迅速化に向けた計画が策定済	一部商品で迅速な開発・提供が可能	システム制約なく商品開発できる環境が実現	
	データガバナンス	データ形式や品質に関する課題が未整理	データ品質問題を認識	データ標準化と品質管理の計画が策定済	主要なデータの標準化・統合が進行中	全社的なデータ品質管理体制が確立済	
移行リスク管理	リスク評価体制	移行リスクは未評価	主要リスクが特定済	詳細なリスク管理計画が策定済	リスク対策が実装・検証済	継続的リスクモニタリングと対応実施済	

	評価項目	レベル1 初期段階	レベル2 認識段階	レベル3 計画段階	レベル4 実行段階	レベル5 最適化段階	自社 評価
移行リス ク管理	事業継続 計画	移行中の事 業継続計画 は未検討	移行中の事 業継続リス ク特定済	詳細な事業 継続計画が 策定済	事業継続計 画がテスト 済	完全な無停止移 行体制が確立済	
	コスト管 理計画	移行コスト の見積りが ない	概算コスト が見積済	詳細な予算 と ROI 分析 が完了	コスト管理 体制が確立 済	予算内で最大の 価値を実現	
クラウド ネイティ ブ文化と 開発手法	DevOps 文化の定 着	開発と運用 は完全に分 離	DevOps の 必要性を認 識している	一部チーム で DevOps 手法を試験 導入	主要システ ムで DevOps 体 制が確立	全社的に DevOps が定着 している	
	アジャイル開発の 採用	ウォーター フォール型 開発のみ	アジャイル の利点を認 識	一部プロジ ェクトでア ジャイル導 入	主要商品開 発でアジャ イル手法を 活用	全社的にアジャ イル文化が定着	
	自動化の 推進状況	手動操作が 中心の運用	一部プロセ スの自動化 検討	CI/CD パイ プラインの 構築計画策 定済	主要システ ムで自動化 実装済	全社的な自動化 体制確立	
データ戦 略とガバ ナンス	データガ バナンス 体制	データポリ シー未整備	データガバ ナンスの必 要性認識	データガバ ナンスフレ ームワーク 策定中	主要データ に対するガ バナンス体 制確立	全社的データガ バナンス体制完 備	
	アナリテ ィクス活 用	基本的な集 計分析のみ	高度分析の 価値を認識	分析環境の クラウド移 行計画策定 済	一部業務で クラウド分 析基盤活用 中	全社的なデータ ドリブン意思決 定実現	
	保険デー タセキュ リティ	伝統的な境 界防御中心	クラウドセ キュリティ 課題を認識	クラウドセ キュリティ 戦略策定中	ゼロトラス ト等新セキ ュリティモ デル導入済	最新のセキュリ ティモデル最適 化	

評価方法と活用ガイド

（１）評価プロセス

- ・各項目をレベル 1～5 で自己評価
- ・部門横断チームによる評価を推奨
- ・四半期または半年ごとに再評価

（２）スコアリング

- ・全体平均スコア = 全項目の平均値
- ・領域別スコア = 各領域の平均値
- ・ギャップ分析 = レベル 5 との差分を項目ごとに分析

（３）経営判断への活用

- ・平均スコア 2.0 未満: 基礎構築フェーズ（意識改革と教育が優先課題）
- ・平均スコア 2.0～3.0: 計画策定フェーズ（詳細計画の策定と初期実装が優先課題）
- ・平均スコア 3.0～4.0: 実行フェーズ（計画的実行と成果測定が優先課題）
- ・平均スコア 4.0 以上: 最適化フェーズ（継続的改善と価値最大化が優先課題）

（４）優先課題の特定

- ・最もスコアが低い領域を重点的に改善
- ・依存関係のある項目をグループ化して対応
- ・短期的成果を出せる項目から着手

謝辞

本論文を執筆するにあたりベンダーレクチャーを実施頂いた、アマゾン ウェブ サービス ジャパン合同会社様、株式会社日立製作所様、日本アイ・ビー・エム株式会社様、並びにアンケートにご協力いただきました日本アクチュアリー会法人会員各社様には、心よりお礼を申し上げます。

参考文献

- 野村友規,『GitHub CI/CD 実践ガイド』、技術評論社,2024
- 山田 明憲,『Docker/Kubernetes 実践コンテナ開発入門』,技術評論社,2024
- 米久保剛,『アーキテクトの教科書』、翔泳社,2024
- José Haro Peralta,『実践マイクロサービス API』,翔泳社,2023
- Mark Richards & Neal Ford,『ソフトウェアアーキテクチャの基礎』、オライリー, 2022
- Sam Newman,『モノリスからマイクロサービスへ』、オライリー, 2020
- David Scott Bernstein,『レガシーコードからの脱却』、オライリー, 2019
- Martin Fowler,『リファクタリング(第2版)』、オーム社, 2019
- 黒山通彦・平山智晴・松本卓也・片山博順『マッキンゼーが解き明かす生き残るための DX』 日経 BP,2021
- デジタルトランスフォーメーション D X レポート ～IT システム「2025 年の崖」の克服と DX の本格的な展開～-経済産業省,
https://www.meti.go.jp/shingikai/mono_info_service/digital_transformation/pdf/20180907_03.pdf
- TIOBE Index - TIOBE,
<https://www.tiobe.com/tiobe-index/>
- COBOL blues - REUTERS GRAPHICS,
<https://www.reuters.com/graphics/USA-BANKS-COBOL/010040KH18J/>
- 保険会社が選択すべき I T インフラの研究 - 日本アクチュアリー会,
https://www.actuaries.jp/lib/IT_ronbun/pdf/IT2023-1.pdf

- 2024 年保険業界のグローバル展望 - Deloitte,
<https://www2.deloitte.com/content/dam/Deloitte/jp/Documents/financial-services/ins/jp-ins-2024-global-insurance-outlook.pdf>
- IBM が生保業界の生殺与奪を握る理由、DX どころではない「レガシーシステム」の超難題 - ダイヤモンド・オンライン,
<https://diamond.jp/articles/-/297952>
- 生命保険会社におけるパブリッククラウドの活用～機動的なシステムインフラ構築とコスト最適化を目指して～-情報処理学会,
<https://www.ipsj.or.jp/dp/contents/publication/47/S1203-U02.html>
- サステナビリティを踏まえた保険業界における DX 推進 - アクチュアリー会,
https://www.actuaries.jp/lib/IT_ronbun/pdf/IT2021-2.pdf
- 生命保険会社におけるパブリッククラウドの活用- 情報処理学会,
https://jp.fujitsu.com/family/article_m/2020/pdf/2020-04.pdf
- 生命保険システム-Fintage,
<https://www.finbridge.jp/overview/overview-life-insurance/>
- 「枯れたシステム」メインフレームが今も大量に稼働している理由…根強い需要 -ビジネスジャーナル,
https://biz-journal.jp/company/post_385624.html
- データセンター業界レポート～データセンター業界の最新の動向～ | 調査研究レポート -日本政策投資銀行（DBJ）,
https://www.dbj.jp/topics/investigate/2021/html/20211122_203598.html
- 企業 IT 動向調査 報告書 2024- 一般社団法人日本情報システム・ユーザー協会（JUAS）,
https://juas.or.jp/cms/media/2025/01/JUAS_IT2024.pdf