API の活用に関する実践状況調査

概要報告書

2022 年 10 月 26 日

独立行政法人情報処理推進機構
社会基盤センター

IPA 独立行政法人情報処理推進機構
社会基盤センター

1. 調査概要（背景、目的、実施内容等）

1.1. 背景・目的

あらゆる産業において、新たなデジタル技術を使ってこれまでにないビジネスモデルを展開する新規参入者が登場し、実際にゲームチェンジが起こっている。こうした中で、各企業は、競争力維持・強化のために、デジタルトランスフォーメーション（DX：Digital Transformation 以下「DX」という。）をスピーディに進めていくことが求められている。しかし現実には、我が国産業界における DX の推進は「DX の取り組みを始めている企業」と「まだ何も取り組めていない企業」に二極化しつつある。2020 年に経済産業省が発表した、「DX レポート 2 中間取りまとめ」によると、依然として 9 割以上の企業が、DX にまったく取り組んでいないレベルにあるか、DX の散発的な実施に留まっているに過ぎない段階である傾向が明らかになっている。

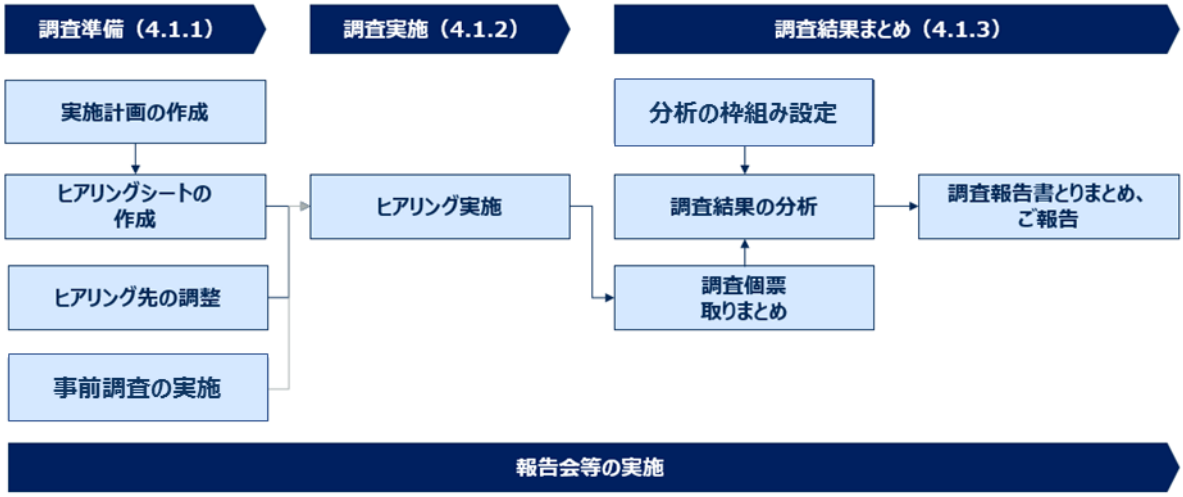
情報処理推進機構（以下「IPA」という。）では、企業における DX を促進するための取り組みの一つとして、これから DX の実践に向けて取り組み始める、もしくは取り組んでいる最中にある組織の DX 推進の担当者が、DX の実現に向けた取り組みを実践する際に参照するものとして、DX 実践手引書 IT システム構築編（以下「DX 実践手引書」という。）を作成している。全 4 章の構成で、DX を実現するための考え方をまとめた第 1 章、DX を継続的に進めるための考え方をまとめた第 2 章、DX を実現するための IT システムのあるべき姿をまとめた第 3 章、あるべき IT システムとそれを実現する技術要素をまとめた第 4 章を公開した。そして「第 4 章：あるべき IT システムとそれを実現する技術要素」に含まれる「4.5. API(Application Programming Interface)」（以下「API」という。）の活用について拡充を行うため、新たな IT システムの設計開発手法の検討を進める上で参考となる API の役割、管理の仕方、運用方法などの情報や事例をまとめるため、調査・整理・分析を行った。

本調査は、DX 実践手引書において、DX について、これを支える技術的な側面としての API からアプローチし、企業の DX 推進の担当者にとって API を活用するために必要になる考え方や API 活用における要件、考慮すべき点を示すために必要な事例を調査するものである。DX 実践手引書「4.5.API」は、国内の API 活用事例を調査した結果に基づいて、これから DX を推進してゆく組織、または現在組織内で DX を推進していて更なる発展を目指す組織のために、組織内の DX を推進する担当者を想定読者として、DX の取り組みを、API を活用して実現させるための手引きとなるものである。

本調査では、上記の目的を達成するために、先進企業が実際にどのように API を活用しているか、ケーススタディとして調査した。調査においては、最初に企業の API 活用領域、API 活用を取り巻く関係者、API を活用する背景・理由など、企業での API 活用の全体像を把握した。その上で、API または API 群の設計思想・方針、非機能要件の担保方法、セキュリティ、開発要件、データの利用・活用について調査した。また、API 全体管理として、API 全体の開発要件、運用要件を調査した。調査においては、企業の API 活用の実態や具体的な取り組みを把握するとともに、企業が特に工夫している点、考慮している点を確認していった。具体的な取り組みの把握にあたっては、取り組みの背景や理由、取り組みによる成果や効果もあわせて確認し、全体像がわかるよう調査した。調査の結果は、DX 実践手引書に、それぞれのケーススタディについて参考にすべき読者を想定した上で、取り組み内容をまとめた。また、工夫している点や考慮している点のスサノオ・フレームワーク領域による違いを、API 活用領域別の特性として整理した。

1.2. 実施内容

DX 実践手引書に記載する API 活用に関する取り組み事項や考慮すべき事項を想定して、各企業へのヒアリング項目・観点を定めた。また、ヒアリング項目・観点の妥当性やマクロ視点での企業の API 活用状況を把握するために、API 管理ツール提供会社に事前調査を実施した。その後、API 活用先進企業へ API 活用に関するヒアリングを実施し、企業ごとの API 活用に関する実践状況をまとめるとともに、有識者の見解を踏まえてヒアリング結果を整理・分析した。



1.3. ヒアリング対象企業選定の考え方

スサノオ・フレームワーク領域を考慮して、本調査目的に合致する企業を文献調査、API 管理ツール提供企業からの推薦を経て、候補企業として選定した。その中から、スサノオ・フレームワーク領域でカバーしている API 活用領域や API 活用内容を吟味・検討した上で、ヒアリング依頼企業を決定した。その後、ヒアリング調査に応じて頂ける企業を対象に、ヒアリング調査を実施した。

1.4. ヒアリング実施企業

本調査目的に合致する 20 社のロングリストからヒアリング調査企業としての妥当性を議論の上、ヒアリング調査に応じて頂ける企業として国内企業 4 社を対象に調査を実施した。また、事業会社の他に、様々な国内企業の事例を知っている API 管理ツール提供企業 1 社を対象に調査を実施した。

	企業名(仮名)	業種	ヒアリングでカバーできたスサノオ・フレームワークでのAPI活用領域					
			A.各社独自サービス	B.業務・基幹システム	D/E.データ活用・分析	H.他業種連携/外部サービス	I.外部共通基盤	J.社外データソース
1	情報・通信業A社	情報・通信	●	●				
2	保険業B社	保険（損害保険）	●	●	●	●	●	●
3	小売業C社	小売	●	●	●	●	●	●
4	製造業D社	製造（電気機器）		●	●		●	●

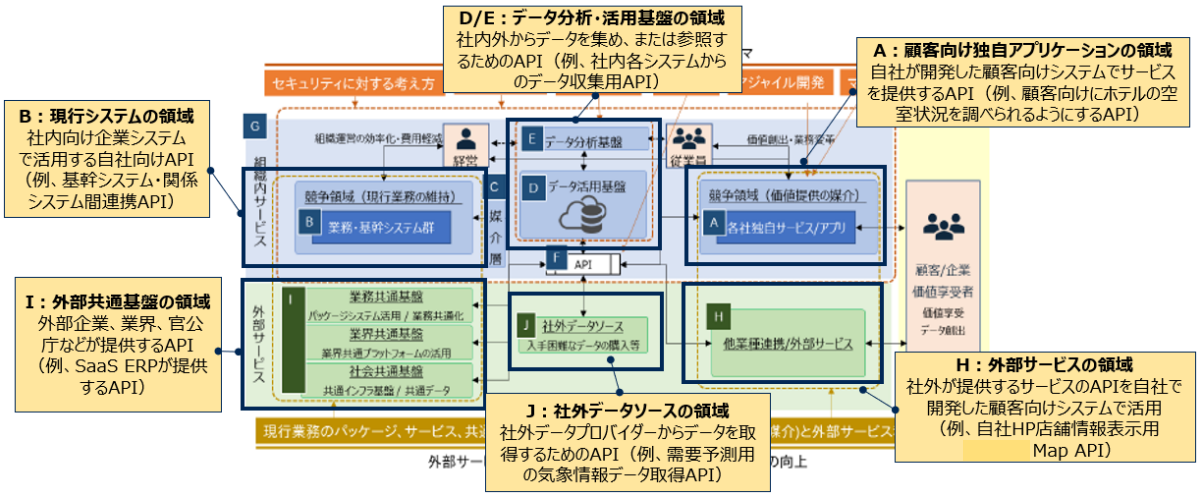
2. 調査結果（各ヒアリング結果の内容）

2.1. ヒアリング調査概要

ヒアリング調査の実施に向けて、ヒアリング項目の洗い出し・整理を行い、28 のヒアリング項目を設定した。28 のヒアリング項目は、API 活用の全体像、API 個別・API 群、API 全体管理の大きく 3 つの観点で構成されている。API 個別・API 群の観点では、さらに「設計思想・方針の有無」、「API の非機能要件担保方法」、「API セキュリティ」、「API の開発要件」、「API を介したデータの利用・活用」の 5 つの分類の観点でヒアリング項目が構成されている。API 全体管理の観点では、「API の開発要件」、「API の運用要件」の 2 つの分類の観点で構成されている。

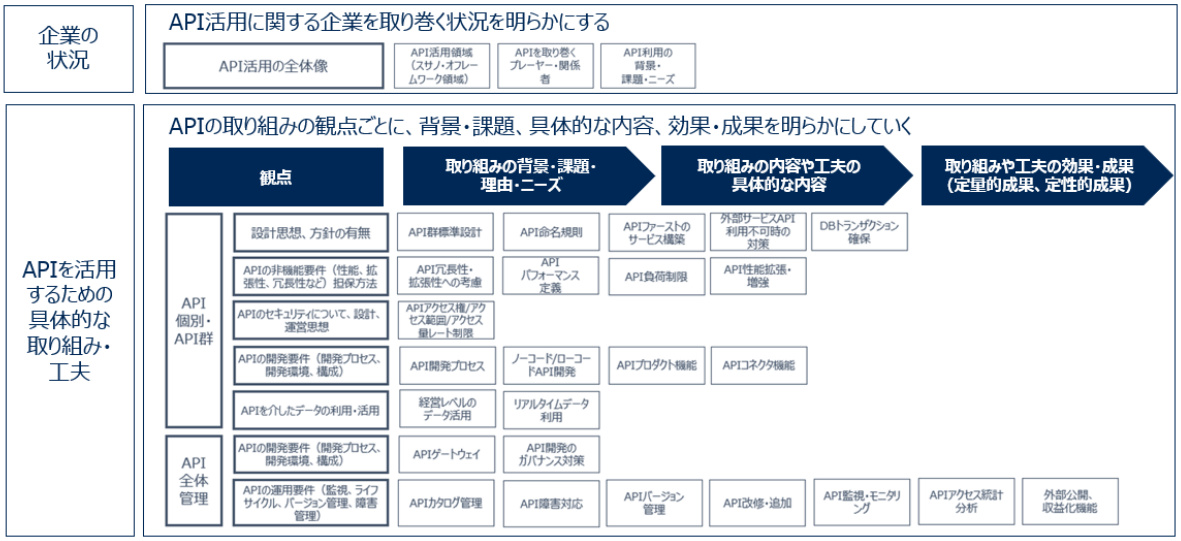
ヒアリング調査の実施においては、最初に API 活用の全体像を把握した上で、API 個別・API 群、API 全体管理それぞれのヒアリング項目ごとの取り組みや工夫の具体的な内容を確認した。その際、取り組みや工夫の内容だけでなく、その背景や理由、取り組みや工夫による効果や成果もあわせて確認した。

API 活用領域については、スサノオ・フレームワーク領域の定義を元に領域ごとの API 事例を定め、ヒアリング調査企業担当者とも情報共有して API 活用領域を容易に特定できるようにした。以下、スサノオ・フレームワーク領域の表現にあたっては、下記の図を元に、組織内サービス領域として、顧客向け独自アプリケーションの領域を「A 領域」、現行システムの領域を「B 領域」、データ分析・活用基盤の領域を「D/E 領域」とする。また、外部サービス領域として、外部サービスの領域を「H 領域」、外部共通基盤の領域を「I 領域」、社外データソースの領域を「J 領域」とする。



2.2. ヒアリング調査結果の企業別の整理・分析方法

今回のヒアリング調査では、ヒアリング対象企業の API 活用に関する状況をまず明らかにした上で、API を活用するための具体的な取り組み・工夫を確認した。その際、取り組みや工夫の内容だけでなく、その背景や理由、取り組みや工夫による効果や成果もあわせて確認した。このやり方に沿って、ヒアリング調査結果を企業別に以下の図のように整理・分析した。



「DX 実践手引書」での事例記載内容を考慮して、ヒアリング調査結果の企業別の整理・分析方法として、以下の記載項目を設定した。

- 立場・読者：　事例が参考になると考えられる想定読者
- スサノオ・フレームワーク領域：　ヒアリング対象企業の「2.1 ヒアリング調査概要」で示した API 活用領域。（顧客向け独自アプリケーション領域で API 活用している場合は「A 領域」など）
- 概略：　読者にとって重要なポイント（価値、提言、留意事項、リスク、キーワードなど）
- 背景・課題・ニーズ：　API 全体の取り組みに関係する企業を取り巻く状況や背景・課題・ニーズ。API を活用するために行った個別の取り組み・工夫・手法の背景、従前の課題やニーズ
- 取り組みとその効果：　API を活用するために行った具体的な取り組みや工夫・手法の内容と、工夫・手法の結果、得られた効果や定量的・定性的成果

上記の記載項目を作成するにあたっては、前述の調査分析の枠組みから以下の図のような対応付けを行った。

2.3. ヒアリング調査結果

1. 情報・通信業 A 社

■ 立場・読者

- API 管理者（利用側）
  - ✓ 多くのサイロ化した基幹システムが存在する中で、顧客向けサービスを短サイクルで開発・提供したいと考えている管理者
  - ✓ API 全体管理・ガバナンスの役割を担っている管理者
  - ✓ API 管理ツールを活用した API 全体管理の仕組みを構築したいと考えている管理者
  - ✓ API 活用を促進するために、開発者向けの環境作りをしたいと考えている管理者
  - ✓ 安定した API 運用を実現したいと考えている管理者
  - ✓ API 管理工数削減の取り組みを考えている管理者

■ スサノオ・フレームワーク領域

- A 領域。様々なサービスで利用している。
- B 領域。決済システムなど基幹システムで利用している。
- 全体管理。API カタログ管理、API 監視・モニタリング、API アクセス統計分析、API ゲートウェイ、API 管理ツール活用などを行っている。

■ 概略

- 様々な社内システムとの連携を API 基盤（API ゲートウェイ）に集約することで、システム管理工数を削減しつつ、顧客向けサービス開発短サイクル化と安定した API 運用・サービス提供を実現した。

■ 背景・課題・ニーズ

- 通信以外の事業・サービスを拡大していく方針のもと、蓄積した顧客・決済情報を社内の様々な事業・サービスに開放して新たな事業・サービス開発を行う際に、個別システム間データ連携をするのではなく、API でシステム連携を行う流れが出てきた。
- 世の中が、オンプレミス開発からクラウド利用の流れになる中で、様々な新しい技術を活用できるようになり、API 基盤を構築して、さらに開発スピード向上や開発コスト削減を目指すようになった。
- 社内に様々なシステム基盤があり、サービス開発を行う際、システム基盤上に存在する仕様の違う API を学習することに多大な時間がかかっていた。

■ 取り組みとその効果

- API(群)個別の取り組み/設計思想・方針の有無

(背景・課題・ニーズ)

- ✓ 基幹システムに存在する独自仕様の API を多大な時間をかけて学習するのではなく、標準化された API 設計仕様にすることにより、API 開発者の API 学習コストを削減したい。
- ✓ B 領域では、基幹システム上に独自仕様の既存 API が多数存在しており、その仕様に慣れた開発者がいる。

(取り組み)

- ✓ API 群標準設計について、1 つは、設計仕様を揃え標準化を進める方針、もう 1 つは、あえて完全には標準化せず、既存 API 仕様をある程度踏襲する方針の 2 種類の設計方針をとっている。前者により API 開発者の学習時間を削減するとともに、後者により既存 API 開発者の学習時間が大幅に増えないようにしている。また、前者の方針の標準化された API と、後者の方針の部分的に標準化された API の 2 つを用意することも進めている。そうすることで、既存 API に慣れ親しんでいない開発者、慣れ親しんでいる開発者の双方にとって、API を利用しやすい状況を作っている。

(効果)

- ✓ 他の取り組みと合わせて、開発者の API 学習時間を 70%以上削減した。
- ✓ サービスリリースを短サイクル化できた。

- API(群)個別の取り組み/ API 非機能要件担保方法

(背景・課題・ニーズ)

- ✓ API 基盤が接続する基幹システムには性能限界があり、API 基盤上の API 連携がその性能限界を超えることで、基幹システムをダウンさせてはいけない。
- ✓ 基幹システム上の API に直接接続していた時と比較して、API 基盤経由で API 接続した場合のパフォーマンスを落とすことはできない。

(取り組み)

- ✓ API 開発のもととなるユースケースを確認して、同じような API が社内に存在しないか確認することで、API 冗長性を防止している。
- ✓ 基幹システム上の API を直接利用していた時のパフォーマンスを基準にしてパフォーマンス閾値を設定している。
- ✓ API 管理ツールを活用して、基幹システム性能を基準に API 負荷制限の閾値を設定して、API 負荷を制御している。

- ✓ API 項目情報追加をトリガーにして、情報追加によって性能が変わることに対応するために API 性能拡張・増強を行っている。

（効果）

- ✓ パフォーマンス計測とチューニング改善により API 基盤経由接続の場合でも、基幹システム API を直接利用していた時と同様のパフォーマンスを出すことができた。
- ✓ 基幹システム側の問題を除き、API 基盤単体としては、3 年間システムダウンなく運営できている。

➢ API(群)個別の取り組み/ API のセキュリティ

（背景・課題・ニーズ）

- ✓ API 基盤は、現時点ではサーバー間のインターフェースをカバーしている。
- ✓ API 基盤接続先の基幹システムは、急激なトラフィック増加に柔軟に対応することはできない。

（取り組み）

- ✓ API 開発者のリクエストに応じて、API 管理ツールが発行する API キーを使わなければアクセスできない仕組みにしている。
- ✓ API 基盤は、現時点ではサーバー間のインターフェースとなっているため、どのサーバーが API 基盤の API を利用しているか管理している。
- ✓ 顧客向けサービス提供側に最大アクセス量を提供してもらい、基幹システム側とトラフィック量を合意し、トラフィック量に収まるよう API を設計している。API 管理ツールのプロダクト機能を利用して、トラフィック制限をしている。

（効果）

- ✓ API 基盤上、API のセキュリティやトラフィックに関する問題を発生させていない。

➢ API(群)個別の取り組み/ API の開発要件

（背景・課題・ニーズ）

- ✓ 社内に様々なシステム基盤があり、サービス開発を行う際には、システム基盤上の個別 API 仕様を考慮してシステム連携開発をする必要があった。

（取り組み）

- ✓ API 管理ツールのコネクタ機能を利用してシステム間連携を容易にしている。
- ✓ 短期間でシステム開発を行うための一般的な取り組みを実施しているが、API 開発に特化した特別な取り組みは行っていない。

（効果）

- ✓ API 管理ツールのコネクタ機能を利用することによりシステム間のつなぎこみが容易になり個別 API の仕様を気にしなくて良くなった。

➢ API 全体管理/ API 開発要件

（背景・課題・ニーズ）

- ✓ 社内に様々なシステム基盤があり、システム間連携が複雑になっていた。API 利用方法を共通化して連携しやすくしたかった。

（取り組み）

- ✓ API ゲートウェイとして API 基盤を構築している。
- ✓ API を開発する際は、API 基盤管理部署が、ユースケースを元に、既存 API で同様なものが存在するか、開発後利用されるかを精査し、優先度を考えて、API 開発可否を決定している。

（効果）

- ✓ 他の取り組みと合わせて、①学習コスト削減 70%以上 ②サービスリリース短サイクル化 ③2 営業日で API 利用可能 ④アプリケーション完成からリリースまでの時間が 15 分 ⑤システム停止なし ⑥IaaS・PaaS 部分はノータッチ運用、といった成果を得られた。
- ✓ API の冗長性に関する問題は発生していない。

➢ API 全体管理/ API 運用要件

（背景・課題・ニーズ）

- ✓ API に関する情報を公開しなかった場合、API 開発者から API 仕様や利用方法について、API 基盤管理部署に多くの質問・問い合わせが来ると想定できた。その場合、本来の仕事である開発稼働時間が減ってしまう。そうならないように、開発者側で自己解決できる仕組みを提供したかった。
- ✓ 顧客向けサービス利用者は何千万人単位であり、サービス要求水準が高く、問題が発生した場合、大きなクレームになる。
- ✓ API 基盤管理部署は、API 基盤構築・運用だけでなく、元々は、顧客向けサービス開発を行っていた。

（取り組み）

- ✓ API カタログ管理を行うために、社内サイト上にすべての API 情報（どういう API があるか、どのようなシークエンスかなど）を掲載し、情報の取得や手続きができるシステムを社内で作り用意している。
- ✓ 監視ツールを活用してリアルタイムに API 状況（トラフィック量、応答速度、エラー件数など）を見られるようにしている。閾値を超えたらアラートを出している。リアルタイム監視とは別に、一週間の傾向からこれまでと違う結果が出た場合、原因などを深掘り調査して、将来的に問題が起きないように先手を打つ対策も行っている。
- ✓ 監視ツールダッシュボード画面を活用して障害分析を行っている。
- ✓ サイト導線など顧客行動の分析も行い、サービス利用増加と API 利用増加の関係などを分析している。

（効果）

- ✓ API カタログ管理や API 情報掲載を行う社内サイトを開発者に提供することで、API 基盤管理部署の API 問い合わせ対応時間が、通常と比較して 10 分の 1 になった。
- ✓ API 基盤は、3 年間システムダウンがなかった。そのため、システムトラブル調査工数が発生しなかった。
- ✓ 普段から特異点を監視することで、問題が発生した場合の調査工数を削減できる。
- ✓ サイト導線など顧客行動を API 利用状況とあわせて分析することで、サービス開発の中長期的な取り組みを進める上でのアイデアを得ることができる。

2. 保険業 B 社

■ 立場・読者

- ➢ API 管理者（利用側）
  - ✓ 既に多くの API を社内システムで利用している中で、API の一元的な管理を行いたいと考えている管理者
  - ✓ API 全体管理・ガバナンスの役割を担っている管理者
  - ✓ API 管理ツールを活用した API 全体管理の仕組みを構築したいと考えている管理者
  - ✓ API の非機能要件やセキュリティを強化したいと考えている管理者
  - ✓ API に関する社外ベンダーへのガバナンスを利かせたいと考えている管理者
  - ✓ API を活用したシステム開発スピードを向上させたいと考えている管理者

■ スサノオ・フレームワーク領域

- A 領域。代理店側システムで保険契約ができるようにしている。
- B 領域。契約管理や保険金の支払い向けシステムなど基幹システムで利用している。
- D/E 領域。全社統一的な DWH を作るようなデータ分析基盤と基幹システム間連携で利用している。
- H 領域。地図情報の API を利用している。
- I 領域。SaaS（Software as a Service）提供 API を利用している。SaaS では、例えば、OCR などで読み込んでもらったデータを整形して送り返してもらうようなサービスで API を利用している。
- J 領域。郵便番号・住所検索や法人データ取得などで利用している。
- 全体管理。API カタログ管理、API 監視・モニタリング、API アクセス統計分析、API ゲートウェイ、API 管理ツール活用などを行っている。

■ 概略

- 幅広い領域での API 活用が増加する中、API ゲートウェイ構築により、ガバナンスを利かせ、一貫した非機能要件やセキュリティを確保しつつ、開発スピード向上と安定したシステム運用を実現した。

■ 背景・課題・ニーズ

- 社内システムがそれぞれ個別に API を利用していたが、周辺システムが増えてくる中で、API に接続するシステムが増えていき、流量制御や利用情報の把握などが難しくなっていった。
- もともと社内システムの個別 API が、様々なシステムで多く使用されるようになることを想定していなかった。
- API 非機能要件やセキュリティ、開発スピードを担保するために API ゲートウェイを構築した。
- 接続方式を標準化し、ガバナンスをかけセキュリティを担保するという欧米での流行・方向性に倣い、API ゲートウェイを構築する流れになった。

■ 取り組みとその効果

- API(群)個別の取り組み/設計思想・方針の有無

(背景・課題・ニーズ)

  - ✓ API に限らず外部委託（ベンダー、SaaS、PaaS、ASP）を行う際は、委託管理チェックをしており、運用開発の分離、個人情報管理レベル、セキュリティなどが、適切なガバナンスのもと運用されているか確認している。

- ✓ API で外部データを取得するシステムを構築する場合は、外部サービスの可用性の妥当性を評価した上で利用している。
- ✓ 基本的には API ゲートウェイ経由で接続する方針を進めており、API ゲートウェイを経由しない接続方法は、以前から存在する API を除いて、例外的な位置づけになっている。

（取り組み）

- ✓ 外部サービス API 利用不可時の対策としては、API が使われるシステムの重要度により、対策を行っている。API の技術的な側面で対策をするのではなく、API を提供する社外ベンダーとの SLA をチェックしている。システム可用性の求めているレベルに応じて都度、ベンダーに求めるものを判断している。
- ✓ データベースのトランザクション確保に関しては、一つのトランザクションは一つの API で処理し、複数の API になる場合は、順序性を担保し制御するようにしている。

（効果）

- ✓ 外部サービスの障害が、重要システムの障害につながるような事象に至ったことはない。

➢ API(群)個別の取り組み/ API 非機能要件担保方法

（背景・課題・ニーズ）

- ✓ API ゲートウェイのプロジェクトが始まった時から、開発メンバーは、非機能要件に留意していた。
- ✓ API 基盤を使用するシステムは、可用性ランクの中でも上位のものが多い。可用性ランクが低いものは、API 基盤にはあまりない。
- ✓ 安定したシステム運用が求められる金融保険業ということもあり、多額の費用をかけてシステムの非機能要件を担保している。

（取り組み）

- ✓ パフォーマンス定義においては、情報を送信元（API 基盤側）と送信先（API 利用システム側）の両方を考慮して設計している。
- ✓ API のアクセス量が増える場合は、API ゲートウェイを増やす。API ゲートウェイも種類によって分散させ、業務種別でサーバーを分けている。想定パフォーマンスに近くなった場合は、スケールアップあるいはスケールアウトするようにできている。送信先（API システム利用側）は、データベースがネックになることがあるので、データベースを大きくするかどうかを毎回検討している。
- ✓ システムの可用性を、ランク付けしている。API 基盤を使用するシステムは、可用性の高いシステムが多い。ランクに応じて非機能要件を

しっかり検討する仕組みになっているので、必然的にパフォーマンス定義をきっちり行っている。

（効果）

- ✓ パフォーマンスに関する問題は発生していない。

- API(群)個別の取り組み/ API を介したデータの利用・活用

（背景・課題・ニーズ）

- ✓ ビジネス領域を横断した統合的なデータを分析できる環境が求められていた。

（取り組み）

- ✓ 各ビジネス部門が分析作業を実施できる環境から全社的な統合データに API を経由してアクセスできるようにした。

（効果）

- ✓ ビジネス領域ごとでのデータ活用を促進させることができた。

- API(群)個別の取り組み/ API の開発要件

（背景・課題・ニーズ）

- ✓ 外部システムからのデータ漏洩を防止するため、API 接続を許可する場合には、接続先システムのセキュリティレベルを確認する必要があった。

（取り組み）

- ✓ 接続先のセキュリティをチェックするための API のチェックリストを工夫している。API のチェックリストは、開発、プレ本番、本番というように 3 段階に分け、それぞれの段階で満たすべき項目の数を変えている。開発の段階では、最終的に満たすべき項目の説明を行い、完成時の品質水準を示すが、その時点では、全てを満たさなくて良いことにしている。プレ本番、本番と開発が進む中で、チェックリストの精度を厳しくしていく。開発段階で全ての項目を満たせれば良いが、いきなり満たせる開発会社もないので、段階的に要求をあげていき、最終的にゴールに到達するようにしている。
- ✓ API チェックリストは、提供するデータの機密性・重要性に応じてチェックするレベルを変えることで作業負荷を軽減させた。チェック項目を満たさない場合は、API アクセスキーを発行しない運用にしている。
- ✓ API ゲートウェイを構築し、管理ツールにて、API コネクタ機能を利用している。

（効果）

- ✓ 開発が試行錯誤で進む場合もあり、いきなり開発要件を高くしすぎると開発できなくなってしまう。最終的な品質水準を見据えつつ、最初は開発のしやすい状況や進め方にすることで、柔軟に開発ができるようになり、開発スピード向上と高品質を両立している。

➢ API 全体管理/ API 運用要件

（背景・課題・ニーズ）

- ✓ API のデータ送信元システム側と送信先システム側で、API のバージョンを上げる必要があるタイミングが違う。
- ✓ API のバージョンが変わるとインターフェースが変わってしまう。

（取り組み）

- ✓ API バージョン管理においては、なるべく API のバージョンを変えないようにしている。
- ✓ API バージョンを変える場合は、変わるシステムの担当者が、周辺システム担当者に連絡して、周辺システム側と同時にバージョンアップ作業・システム変更作業を行っている。
- ✓ API ゲートウェイを経由していない API は、個別にバージョンアップ作業を行っている。数十のシステムに連携する API があるが、とても苦労してバージョンアップ作業を行っている。関係するシステムの漏れがないか不安になる。そのため、API ゲートウェイを通すようにしている。API ゲートウェイを活用することで関係するシステムの漏れがなくなる。

（効果）

- ✓ API ゲートウェイを活用することで、バージョンアップ作業が効率化された。

3. 小売業 C 社

■ 立場・読者

➢ API 管理者（利用側）

- ✓ API を活用して顧客向けサービスを短サイクルで開発・提供したいと考えている管理者
- ✓ API 全体管理・ガバナンスの役割を担っている管理者
- ✓ ミッションクリティカルなシステムで外部サービス API を活用している管理者
- ✓ サイロ化したシステム運用から API でシステム間連携するシステム運用への移行に取り組もうとしている管理者

- ✓ 安定した API 運用を実現したいと考えている管理者

- API 開発者（利用側）
  - ✓ API を開発しやすい環境を構築するための取り組みを考えている開発者
  - ✓ 顧客向けサービス開発者
  - ✓ より効率的に API 開発を行いたいと考えている開発者

■ スサノオ・フレームワーク領域

- A 領域。EC サイトで利用している。
- B 領域。売上・在庫・物流など基幹システムで利用している。
- D/E 領域。データ分析で利用している。
- H 領域。具体的な利用方法について言及はなかった。
- I 領域。SaaS システムとの連携で使用している。
- J 領域。具体的な利用方法について言及はなかった。
- 全体管理。API カタログ管理、API 監視・モニタリング、API アクセス統計分析、API ゲートウェイ、API 管理ツール・性能監視ツール活用などを行っている。

■ 概略

- API 中間層により顧客向けサービスの進化に強いシステム基盤を実現し、さらに非同期 API など新技術活用や管理ツール・性能監視ツール導入で変化に強い柔軟なシステム開発・運用体制を構築した。

■ 背景・課題・ニーズ

- 以前からシステム間連携で API を活用していた。ファイル連携が中心であったが、近年はリアルタイム連携が増えている。フロント側のリアルタイム連携のニーズに対して、基幹システム側をバーストさせるわけにいかないので、API 中間層を設けている。また、基幹システムのデータ変更をフロント側に、ほぼリアルタイムに知らせるイベントソーシングに力を入れている。
- フロント系、バックエンド系それぞれ個別に API を利用している。API の標準化や API 利用・連携状況の可視化を進めるために、API ゲートウェイを構築・拡大している。現状、API 利用回数はわかるが、細かい分析ができない。誰が、どの API を、どのくらい使っているか、わかるようになると、一定値以上は、API を呼ばないようにするなど、バーストを防ぐ対策を行うことができる。また、将来バックエンド側の基幹システムに再構築があった場合でも、フロント側には影響がないなど、二次的な効果

を次のステップで狙っている。

■ 取り組みとその効果

- API(群)個別の取り組み/設計思想・方針の有無

(背景・課題・ニーズ)

- ✓ やるべき EC サイトは、ビジネスの変化にあわせて、ずっと変わってきた。変化対応していくことに対するビジネスニーズが高い。
- ✓ 外部サービス API が、ミッションクリティカルな部分も担っている。
- ✓ データの名前とその中身が、文脈によって変わる。システムがサイロ化しているため、命名の仕方がバラバラになっている。過去に何度もチャレンジしているが、なかなかうまくいっていない。

(取り組み)

- ✓ API ファーストのサービス構築として、EC については、以前から SOA（サービス志向アーキテクチャ）に則ってフロントエンドとバックエンドの分割を行っていた。
- ✓ 外部サービスが利用不可になった場合の影響を事前に可視化して準備しておく。性能監視ツールを導入して、関係者間で問題の共有ができるようにしている。
- ✓ 既存 API に関しては、コストとメリットを勘案して徐々に命名規約を標準化していっている。API 中間層上の新規 API は、命名規約の標準化を進めている。基幹システム側で、データ名の整理を行うことは難しいので、API 中間層上でその点を吸収しようとしている。その上で、将来基幹システム側を再構築することがあれば、名称を新しいものに合わせることを考えている。命名規約の統一は、数年単位の取り組み事項と考えている。

(効果)

- ✓ バックエンドシステムを変更しなくてもフロントエンドシステムを変更できるので、ビジネス変化に柔軟に対応できた。
- ✓ API 連携は、閉じたシステムではないので問題が複雑化しやすく、性能問題が関係システムに波及する。サイロ化された閉じたシステムに慣れているシステム人材の考え方を変革していくことは、難しい。
- ✓ 外部サービス API が利用不可になった場合に、関係システム担当者を含めて、慌てず迅速に障害対応できるようになった。
- ✓ API 命名規約を整備することで、API が再利用しやすくなり、開発品質も向上する。

- API(群)個別の取り組み/ API 非機能要件担保方法

(背景・課題・ニーズ)

- ✓ 可用性の高いシステムでパフォーマンス上の問題が発生すると、ビジネス上の影響が大きい。
- ✓ 顧客のアクセス特性を、完全に性能試験で事前に再現するというのは、不可能な世界になってきている。

(取り組み)

- ✓ 可用性が高いシステムを中心に、パフォーマンス定義を行っている。それ以外のシステムは、個別に行っていて、全体的な管理や可視化はできていない。また、パフォーマンス対策として、レスポンスの悪い API は、非同期化している。
- ✓ 事前のパフォーマンステストでは想定できない部分も大きい。事前に対処する部分と共に、実際に問題が起こった場合に、早く問題を発見し対処することや、起こった問題を分析してシステム担当者で共有し学ぶことで、次に活かすことに取り組んでいる。

(効果)

- ✓ API を非同期化することで、システム間が疎結合になり、耐障害性が向上した。
- ✓ 問題が起こった場合の初動が早くなった。また、問題を分析・共有することでシステム部門内にノウハウをためることができた。

- API(群)個別の取り組み/ API のセキュリティ

(背景・課題・ニーズ)

- ✓ 顧客情報は、トップレベルに重要な情報で、守秘できなかった場合に、法的な問題になる。API 化されることや、イベントストリームにデータが流れ始めるようになると、様々なところで、情報が使えてしまう状態になる。
- ✓ 以前は、大きなプロジェクトの中で、セキュリティ設計をきっちりやれば良かったが、今は小さいチームで開発を進める世界になっている。各チームにきちっとすべてのセキュリティ対策を万全にしてもらうことは、限界がある。

(取り組み)

- ✓ API 管理ツール上で API アクセス権設定やアクセス量レート制限を行うことができるが、まだできていない。
- ✓ インフラレベルと、API レベルの許可認証の 2 系統でセキュリティ対策を行っている。

- ✓ ゲートウェイ経由でしかつなげないようにするなど、設計の議論をしているところで、実装はまだできていない。
- ✓ セキュリティはAPI側で考えて、API利用の問い合わせがあった時、どういう利用意図か確認して、意図に問題がなければ、データを提供するという形でコントロールしたいと考えている。

（効果）

- ✓ 現在取り組みを進めているところであるが、セキュリティ上の問題は発生していない。

➢ API全体管理/ API開発要件

（背景・課題・ニーズ）

- ✓ 歴史的に情報システム部が、ガバナンスをしっかり行っていた。
- ✓ 顧客関係や物流においては、ビジネススピードが速くなっており、リアルタイムにデータ連携する価値が高くなっている。

（取り組み）

- ✓ 各システムで使用されているAPIに関しては開発プロセスのコントロールの中で、ガバナンスをかけている。
- ✓ API中間層上のAPIについては、API中間層の管轄チームがAPIのガバナンスを行っている。
- ✓ 基幹システム上で変化があった時に、その情報をフロント側システムに伝える非同期ストリーミングに取り組んでいる。gRPC(リモートプロシージャコール)や常時接続型は、時期尚早と考えている。

（効果）

- ✓ APIのガバナンス上の問題は発生していない。
- ✓ ECサイト上で、現在店頭で販売している商品在庫を引き当てる仕組みになっている。受注した場合、すぐに店頭で商品確保できるよう、APIでほぼリアルタイムな連携をしている。その結果、在庫回転率向上や販売機会増加を実現している。

➢ API全体管理/ API運用要件

（背景・課題・ニーズ）

- ✓ 今までシステム担当者は、サイロ化したシステムで仕事をしてきたので、他システムの状況に通じていない。APIに関する情報をしっかり共有しないと、APIを理解できない。
- ✓ APIを横断的に使う時代になっていると考えているが、まだ一歩目も踏み出せていないのが現実だと認識している。

（取り組み）

- ✓ API 管理ツールを活用して API カタログ管理を行っている。API 定義を登録するとポータルサイトにカタログ登録されるようになっている。
- ✓ 現状、カタログ管理できている API は、API 全体の一部になっており、できるものから、移行している。
- ✓ システム横断で共通に使い API をカタログ管理することが重要だと考えており、まずは、どの API が誰に使用されているかを把握することが、最初のステップであると考えている。

（効果）

- ✓ API に関する情報をカタログ管理することで、社内で横断的に API を利用できるようにしている。API の再利用促進、API の冗長性を防ぐこと、開発スピード向上を目指している。

4. 製造業 D 社

■ 立場・読者

- ➢ API 管理者（利用側）
  - ✓ ツールを活用して開発・運用の生産性向上と品質向上を実現したいと考えている管理者
  - ✓ API に関するツールの機能を知りたいと考えている管理者
  - ✓ 開発の内製化を進めたいと考えている管理者

- ➢ API 開発者（利用側）
  - ✓ 開発・運用の生産性向上と品質向上を実現したいと考えている開発者
  - ✓ ツールを活用して開発を効率化したいと考えている開発者

■ スサノオ・フレームワーク領域

- ➢ B 領域。基幹システム、会計システムなどで利用している。
- ➢ D/E 領域。マスタデータマネジメントで利用している。
- ➢ I 領域。SaaS システムで利用している。
- ➢ J 領域。マスタデータ品質向上のための外部データ取得で利用している。
- ➢ 全体管理。ツールを活用して API プロダクト機能、API コネクタ機能、API ノーコード開発、API カタログ管理、API ゲートウェイ、API バージョン管理などを行っている。

■ **概略**

- API に関するツールの機能を幅広く活用することで開発・運用の生産性と品質を大きく向上させ、API を活用してデータと業務プロセスを連携させ DX を推進している。

■ **背景・課題・ニーズ**

- DX のケイパビリティを上げていく一つの要素としてデータの統合が課題としてあり、その一つの手段として API があると考えている。業務プロセスをつなげる、統合することで業務効率や生産性を上げるキー要素として API があると考えている。
- データを API でインテグレーションし、マスター・トランザクションデータを入れてグローバルに展開できるデータハブを作り、必要なところに API を通じてデータ配信している。

■ **取り組みとその効果**

- API(群)個別の取り組み/設計思想・方針の有無

(背景・課題・ニーズ)

- ✓ API が、どのアプリケーションで使われているかを重視して管理していた。
- ✓ データを整流化し、データを活用できる状態にすることに力を入れている。

(取り組み)

- ✓ マスターの品質管理のところで、外部 API を使用しており、緊急性を求めていないため、外部サービス API 利用不可時の対策は、特に行っていない。
- ✓ API の命名規約は、アプリケーション名やデータタイプ名、データをイメージできるオブジェクト名を付けるようにしている。API の項目名称は、中身のデータ内容が、複数 API 間で一致するように注意している。

(効果)

- ✓ きれいなマスタデータを保持することができている。

- API(群)個別の取り組み/ API を介したデータの利用・活用

(背景・課題・ニーズ)

- ✓ データ分析ソフトは、グローバル全体で数千ライセンスの規模で利用している。

- ✓ ビジネス側からAPIのレスポンスの速さを求められている。例えば、秒単位でコスト削減をしている生産現場システムと基幹システムとの連携においてニーズが高くなっている。

(取り組み)

- ✓ 各システムからデータハブにデータを持ってくるところは、API連携している。データハブからデータ分析ソフトや経営向けレポートのデータレポジトリーへの連携は、現在はファイル連携している。今後、機械学習で作成した予測モデルの結果とデータ分析ソフトとの連携を、ファイル連携からAPI連携に変更して自動化しようと考えている。
- ✓ APIを活用する意味として、リアルタイム性を考慮している。

(効果)

- ✓ グローバル全体で、ビジネスでのデータ活用が進んでいる。

➢ API(群)個別の取り組み/ API開発要件

(背景・課題・ニーズ)

- ✓ 基幹システム側のエンドポイントに負荷をかけるわけにはいかないのでAPIパフォーマンスに注意する必要がある。
- ✓ APIを色々なところに接続するので、APIのセキュリティの管理をしっかり行う必要がある。
- ✓ API開発は、外部ベンダーではなく、社内IT部門で行っている。

(取り組み)

- ✓ APIプロダクト機能として、統合ツールが提供している機能を利用してアクセス権制御、IPアドレスのホワイトリスト、トラフィック制限の制御を行っている。一元的に管理して、利用状況やパフォーマンス測定、どのユーザーが、どのAPIを、どれぐらい使っているか、などの統計情報もツールで見ることができる。
- ✓ 何千というコネクタを持つAPI管理ツールを使用して、コネクタの接続をおこなっている。
- ✓ 統合ツールを活用して、ノーコード開発を行っている。

(効果)

- ✓ ツールを活用することにより、システム開発運用の品質と効率が向上した。
- ✓ IT部門の新人が、APIノーコード開発を行っている。発工数は3分の1になった。開発の生産性を高めつつ、高品質を実現している。

- API 全体管理/ API 開発要件

（背景・課題・ニーズ）

- ✓ ビジネス側からの要求が多く、システム開発サイクルを短くする必要がある。ビジネスの変化に対応できるよう、API をメンテナンスしやすい状態をつくることが、求められている。

（取り組み）

- ✓ 統合ツールの機能を利用して、API ゲートウェイを構築している。流量制御と権限制御を行っている。
- ✓ 現在、API 全体の 3 割程度が API ゲートウェイ経由になっている。
- ✓ 新しく構築する API は、API ゲートウェイで管理するようにしている。既存 API については、ビジネス要件の優先順位に応じて移行している。開発サイクルを短期化する必要がある API を優先している。
- ✓ API のガバナンス対策としては、API 開発を開放していない。また、開発のガイドラインを定めている。開発環境と本番環境を分離し、定められた手順でテストの確認や承認されたものしか、本番環境にリリースできないようになっている。本番環境へのリリースは、ツールを利用している。本番環境へのリリースは、IT 部門がコントロールしている。
- ✓ API 新技術への対応は、現在は特に行っていない。

（効果）

- ✓ ビジネス側のニーズに応えて、開発サイクルを短期化できた。

- API 全体管理/ API 運用要件

（背景・課題・ニーズ）

- ✓ 今後、API の利用者が増えていく中で、API 情報を共有する手段が必要になってきている。
- ✓ 他のシステム障害は、セキュリティセンターでセキュリティログを一元管理できるようになっているが、API は、一元監視できるようになっていない。API が成功したか失敗したかをモニタリングしていて、エラーが発生した場合は、ログを見て対応している。
- ✓ 開発・運用の品質向上とスピード向上は、常に求められている。

（取り組み）

- ✓ API カタログ管理として、統合ツールの機能を利用して、API を作成した時に、カタログが自動的に作られるようになっている。
- ✓ 障害対応としては、他と同様にエラーを切り分けて、一つ一つ調査している。API は他のシステム障害と比較して、システム間でつながっているため、原因がわかりづらい。システムを順に追って、一つ一つ確認していくしかない。

- ✓ 統合ツールの機能を利用して、API のバージョン管理を行っている。API を使用しているシステムをレポートで確認することができ、バージョンアップの影響範囲も把握することができる。

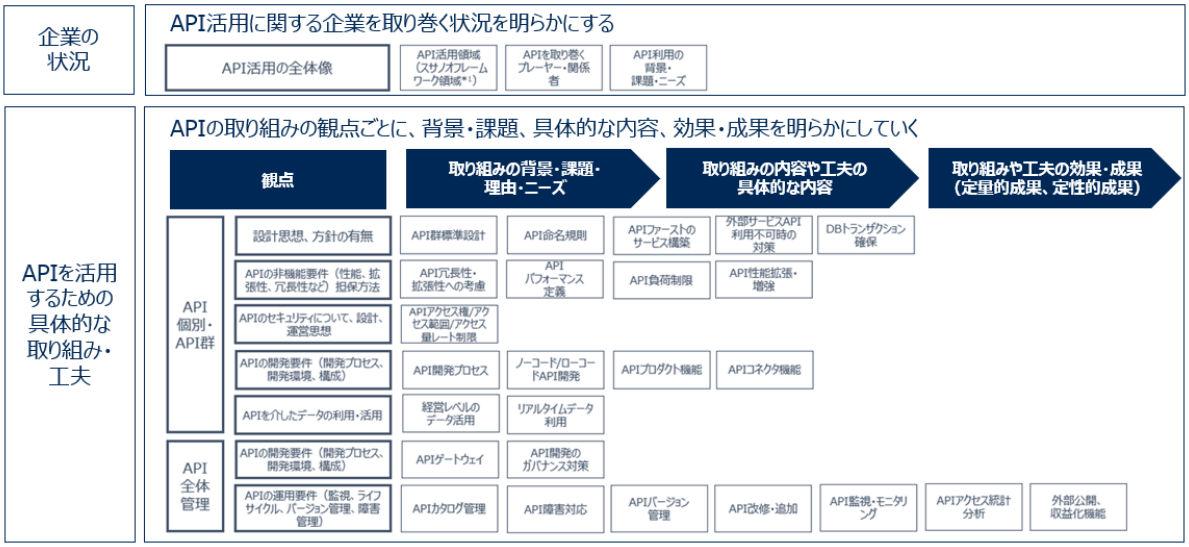
（効果）

- ✓ API カタログ管理をすることで、API の情報を調べやすくなる。現在は、テスト的に取り組んでいる。
- ✓ API の障害監視・一元管理は、今後の取り組み事項になる。
- ✓ ツールを利用することで、開発・運用の品質向上とスピード向上を実現できた。ツール活用前と比較して大幅に工数を削減できた。人依存にならず、自動化して運用できるようになった。

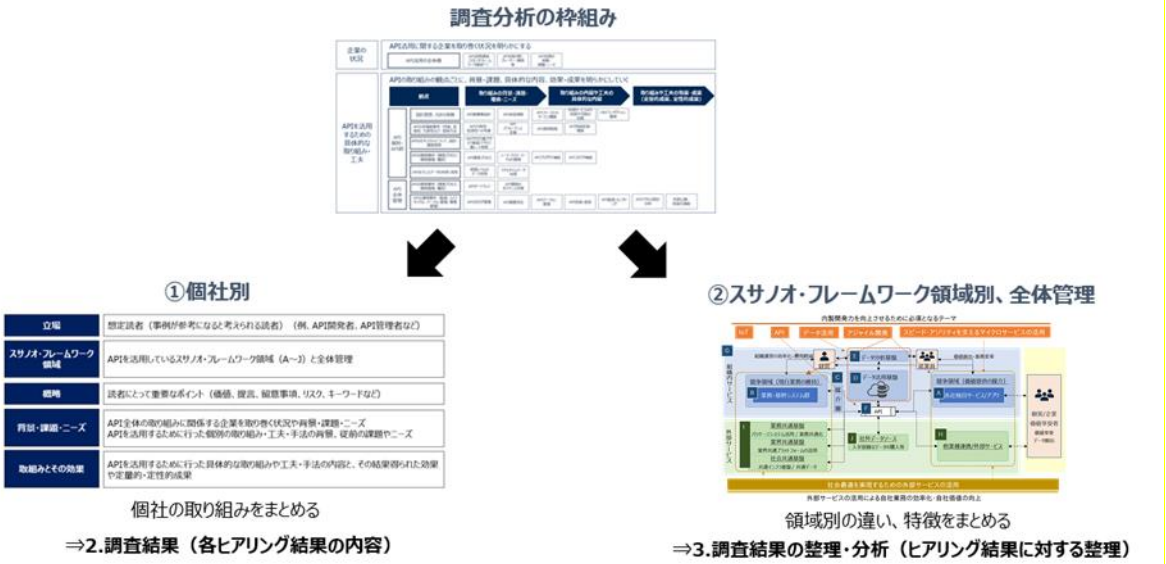
3. 調査結果の整理・分析（ヒアリング結果に対する整理）

3.1. 調査結果の整理・分析方法

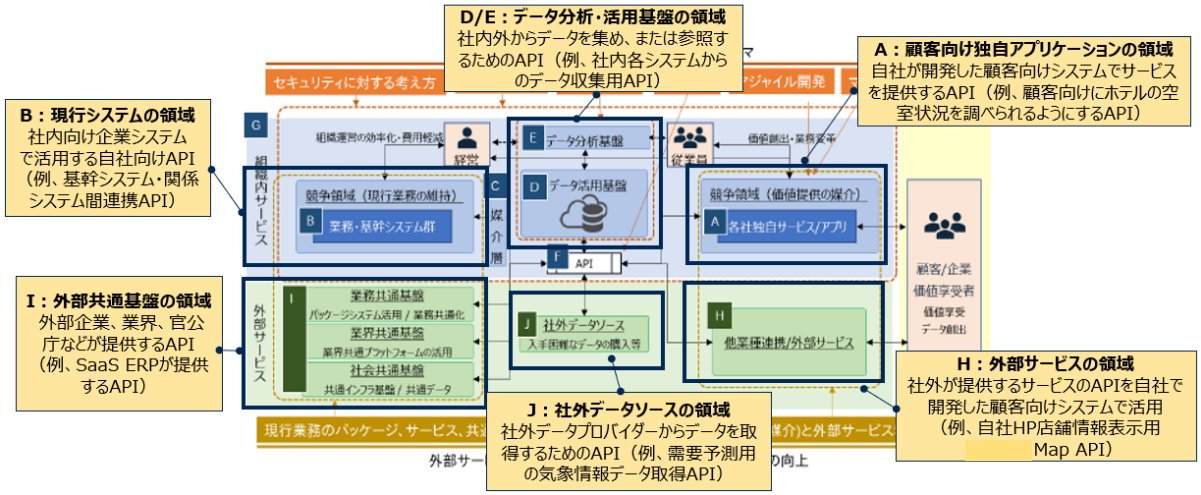
今回のヒアリング調査では、ヒアリング対象企業の API 活用に関する状況をまず明らかにした上で、API を活用するための具体的な取り組み・工夫を確認した。その際、取り組みや工夫の内容だけでなく、その背景や理由、取り組みや工夫による効果や成果もあわせて確認した。



調査結果の整理・分析を、大きく 2 つの切り口で行った。1 つは、ヒアリングを実施した企業別の切り口。もう 1 つは、ヒアリングを実施した企業を横串で整理・分析するスサノオ・フレームワーク領域・全体管理別の切り口である。前者は、「2. 調査結果（各ヒアリング結果の内容）」にて整理・分析した。一方、後者は、本章「調査結果の整理・分析（ヒアリング結果に対する整理）」にて、スサノオ・フレームワーク領域別に、それぞれの特性や領域ごとの違いがあるかないか、を整理・分析した。また、全体管理の特性もあわせて、整理分析した。



3.2. 調査結果の整理（スサノオ・フレームワーク領域別・全体管理）



全体として組織内サービス領域（「A 領域」「B 領域」「D/E 領域」）での API 活用が、外部サービス領域（「H 領域」「I 領域」「J 領域」）と比較して進んでいる。

特に、顧客に新たな価値・サービスを提供する「A 領域」での API 活用が進んでいる。これは、変化の激しい顧客ニーズに対応するためには、提供サービスを短いサイクルでリリース・変更することが必要であり、開発スピードを向上させるためのマイクロサービス化、疎結合なアーキテクチャなどが背景としてある。

1. A 領域

A 領域では、社内開発した API が顧客向けにサービスを提供する IT システム上で活用されている。

DX の流れの中で社内データを活用し社内システムと連携して、顧客向けサービスを充実させる取り組みが最近特に顕著になってきた。顧客向けサービスは変化が激しい傾向があるので、最初から今後 IT システム変更が必要になる状況を前提として、疎結合やマイクロサービスアーキテクチャなどの考え方により API を開発・活用するようになっている。この領域での取り組みは他領域と比較すると歴史が浅いケースが多いため、既存 IT システム上に存在していた既存 API を改修して使用するのではなく、新規に「標準化された API」を開発して使用する考え方が優勢である。

顧客向けサービスが今後増えていくにつれて、開発・利用する API も増えていくことから API 基盤（API ゲートウェイ）を構築して、API 仕様を標準化し API 管理ツールを活用して API 接続を容易にする取り組みや顧客向けサービスのアクセス量をコントロールする取り組みが行われている。

また、ヘッドレスコマースなど、API ファーストの IT システム構築も行われており、顧客体験を重視して、画面(UI)の見た目や操作性を良くする取り組みなどが行われている。

2. B 領域

B 領域では、基幹システムや社内システム間連携に社内開発した API が活用されているケースがある。DX の流れで顧客向けサービスを拡充する取り組みが増えてくる中、社内システムに保存されているデータの活用や、社内システムが持っている IT システムの機能を社外顧客にも提供することが必要となるため、B 領域で API を活用する事例が増えている。

B 領域の特徴としては、社内システム間連携で API を以前から活用していたことがあげられる。以前から活用している API が存在する場合は、IT システムごとに仕様が違っているケースがあり、API を介して直接 IT システム同士が連携する方式であるため、様々な API のトランザクション量やセキュリティのコントロールが難しくなり、個別仕様を学習して開発することに多くの時間がかかる問題が発生している。そのため、API 基盤（API ゲートウェイ）を構築し、API 管理ツールを導入して、B 領域の各 IT システムとの API 連携を標準化し、接続を容易にする取り組みや、非機能要件やセキュリティ要件の設定・コントロールを一元的に行う取り組みが行われている。

3. D/E 領域

D/E 領域では、データを分析・活用するためのデータ分析基盤、データ活用基盤で社内開発した API が活用されている。

ビジネスの意思決定や日常業務でデータをより活用する動きが強まっている。A 領域・B 領域にある社内システムのデータ、I 領域の SaaS システムのデータ、H 領域・J 領域で社外から収集したデータなどを集約し、データを取り出しやすくするためのデータ活用基盤や、データを分析しやすくするためのデータ分析基盤を構築する事例が増えている。様々な IT システムにあるデータをデータ活用基盤に集約する API 上でデータ編集を行っており、データ分析基盤で使用されるデータ分析ツールの情報源となるデータを供給するために API が活用されている。

経営向けのダッシュボードレポートにデータを供給する際に API が活用されるケースもあるが、どちらかというと業務部門が自分たちでデータ分析ツールを利用して分析する際に、データを供給する役割を API が果たしていることが多い。

4. H 領域

H 領域では、社外サービスプロバイダーが開発した汎用的な API を活用して、A 領域の自社開発 API と共に顧客向けサービスを提供している。顧客向けサービスを提供する際には、自社で API を開発するのではなく他社が開発した API（例えば、地図 API など）を利用することで開発工数を削減することや、社内に保持していないデータを活用したサービスを提供するために社外開発 API を活用する事例がある。

H 領域は社外で開発された API であるため、非機能要件を、社外サービスプロバイダーに依存する部分が大きい。しかし、ただ社外サービスプロバイダーに依存するだけでなく、API の監視ツール上で社外開発 API のパフォーマンスを監視し、ボトルネックの発見に努め、パフォーマンス改善活動を実施する取り組みや社外サービスプロバイダーとの SLA 締結時にセキュリティ要件や社外開発 API が利用できなくなった場合の対応を必要に応じて社外サービスプロバイダーと交渉している。

5. I 領域

I 領域では、ERP 製品などパッケージシステムや SaaS システムが提供する API を、B 領域の社内システムとの連携で活用している。基幹システムを自社で開発するのではなく、パッケージシステムを利用することや、SaaS システムを利用して特定の業務機能を行うことが、一般的になってきている。パッケージシステムや SaaS システム側は、パッケージシステムや SaaS システムの導入企業を増やすために、導入企業側の社内システムとの連携がしやすい API を提供することが多い。

自社でパッケージシステムや SaaS システムとのシステム連携を構築する場合と比較して、社外開発 API を利用する方が開発スピードを向上させることができる。但し、社外開発 API であるため、API の内部がどのようになっているか詳細を把握することができない。よって、社内システムとパッケージシステム・SaaS システム間で複雑なデータや大量のデータを連携する際に、思ったようなパフォーマンスが出ない場合、パフォーマンス改善のための対処方法が自社では限られ、パッケージシステム・SaaS システム提供ベンダーに対処してもらう必要がある。

また、SaaS システムの性能上限を考慮する必要がある。特にパフォーマンス、ネットワーク障害、機能・サービスが続行できなかった場合のリトライ回数に注意する必要がある。性能上限を超えないために自社でどのような対策ができるか、性能上限に近づいた時にどうするかを、考えておく必要がある。場合によっては、SaaS ベンダーと交渉し、追加費用を支払い、性能上限を上げる対応を取ってもらうことも一つの手段としてある。

6. J 領域

J 領域では、自社内では保持していないデータ、例えば駅前の地価情報、郵便番号住所検索データや法人企業データ、気象情報などを社外データソースから取得するために API が活用されている。以前であれば、社外データソースから購入したデータは、DVD やファイルデータで一括して入手する場合が多かった。最近は、社外データプロバイダーが API を提供する事例が増えてきている。

J 領域また H 領域もそうであるが、外部プロバイダーAPI を活用する前に、社内領域（A 領域、B 領域、D/E 領域）がモノリシックになっている場合に、まず社内領域を API

化することで、将来的に外部 API と連携しやすい状況を作ろうとしている段階にある企業が多い。社内領域を API 化した後に外部 API を活用することが多くの企業でロードマップになっている。

社外から取得するデータが、ビジネス上ミッションクリティカルである場合は、H 領域の API と同様に、API の監視ツール上で、社外開発 API のパフォーマンスを監視し、ボトルネックの発見に努め、パフォーマンス改善活動を実施する取り組みや、社外サービスプロバイダーとの SLA 締結時にセキュリティ要件や社外開発 API が利用できなくなった場合の対応を必要に応じて社外サービスプロバイダーと交渉している。

7. 全体管理

API の全体管理は、技術的な観点と組織的な観点で捉えることができる。

前者の技術的な観点においては、API ゲートウェイを構築して API 全体管理を行う事例が一般的である。API ゲートウェイの構築においては、市販の API 管理ツールを活用するケースが多い。API ゲートウェイ上の API を標準化し、管理ツールが提供するコネクタ機能を活用することで、API を活用する開発者の API 接続設計・開発時間を短縮することができる。また、管理ツール上で、API のアクセス権やアクセス範囲の設定などセキュリティのコントロールや、トラフィック量の制限設定など非機能要件のコントロール、カタログ管理、バージョン管理を、一元的に行うことができる。API ゲートウェイを構築せず、各 IT システムが個別に API 連携している場合、API がどの IT システム間の連携に使用されているかを管理することは、API が増えてくると煩雑になり難しくなる。API ゲートウェイを構築して管理ツール上で、接続する IT システムを確認・管理できるようになることは、API 管理のガバナンスを向上させる上で重要なメリットになっている。

また、API 監視ツールを導入して、API のトラフィック量、応答速度、エラー件数などを監視する取り組みも行われている。API ゲートウェイ上の API を使用する IT システムは、社内業務遂行や顧客へのサービス提供を行う上で、ミッションクリティカルな役割を担っているケースも多い。そのため、API の運用状況や IT システムトラブルを一元的に把握し、迅速な対応ができるようにすることが、IT システム運用上求められている。

後者の組織的な観点においては、API 全体管理の役割を組織内に持たせることが必要になる。API ゲートウェイを構築する場合は、API ゲートウェイの開発・運用を行う組織が、API 全体管理の役割を担うことが多い。

API 設計・開発時には、API ゲートウェイ上の API を利用する IT システム担当者や部署の間に立ち、データの呼び出し側と提供側の要件を確認・調整して API 設計時にパフォーマンス定義を行うことや、API の冗長性を排除するために、社内に同様の API がないかを確認することなどを行っている。また、API 設計の標準化を推進し、標準仕様に沿った設計が行われているかを確認することも API 管理のガバナンス上、重要である。また、API 開発が完了し API をリリースする際に付与する API アクセスキーの発行や管理ツール上での API アクセス権やアクセス範囲の設定を監督して API のセキュリティを担保する役割も担っている。

API 運用時には、安定して IT システム運用を実現するために API の運用監視や API パフォーマンス改善を主導する役割を担っている。また、API 利用者側の視点に立ち API 開発者に API 情報を提供するサイトの運営や API カタログ管理など、API を開発しやすい環境づくりにおいて重要な役割を果たしている。

4. まとめ（調査のまとめ、DX 推進に活かすための課題と対策）

4.1. API 活用の状況

今回合計 5 社のヒアリング調査を行った。ヒアリング調査社数が限られているため、各社のヒアリング調査結果に加え、API に関するマクロな情報・知識・見識を保有している有識者からの情報や意見を加味し、API 活用の現状を以下にまとめる。

◆ **API 活用の全体像**

■ **API を取り巻くプレーヤー・関係者**

少数の API を利用する段階から統合的に多数の API を活用して DX を推進していく段階になると API 基盤・API ゲートウェイを構築する動きが出てくる。その段階に達すると社内 IT 部門の「IT 基盤を管轄するチーム」が「API 基盤を管轄するチーム」として API 基盤・API ゲートウェイ構築・運用で主導的な役割を果たすことになる。また、「API 基盤を管轄するチーム」とは別に、API を活用して顧客向けサービスや社内向けサービスを「企画・設計・開発するチーム」が別に存在している。「API 基盤を管轄するチーム」と「企画・設計・開発するチーム」は連携しながら API 企画・設計・運用や API 開発管理の役割を担っている。API の開発自体は外部ベンダーに依頼するケースが多いが最近では、API 開発自体も内製化していく動きが出てきている。顧客や事業の変化に迅速に対応するために開発スピード向上が求められている。そのため、都度外部ベンダーに開発を依頼するのではなく、自社の顧客や事業に詳しい開発者を社内に置き、臨機応変に社内開発を行うことで開発スピードを向上させている。また、API 管理ツールが提供するノーコード・ローコード開発機能を活用することで、社内開発を推進しやすい状況を作ることができる。

■ **API 活用の背景・理由**

API を統合的に活用しようとする背景としては、開発スピード向上があげられる。企業は顧客ニーズの変化に合わせて顧客向けサービスを新規に提供する、更新していくことが求められている。特に B to C 企業はそのような状況に置かれていることが多い。顧客向けサービスを提供する際には社内に存在するデータや IT システムを活用・連携していくことが必要になる。そのような取り組みを迅速に進めるために API ゲートウェイ構築や API 仕様の標準化を進め社内システムと連携しやすい開発環境を整備することや API 管理ツールを導入してコネクタ機能利用により IT システム接続を容易にすること、ノーコード・ローコード開発で API 開発スピードを向上させる取り組みが行われている。これはデジタルを活用して顧客向けサービスを拡充したいビジネス部門からの開発要求に IT 部門がタイムリーに答えることができないケースが発生しており IT 部門が API を活用して開発スピード向上を図ることを迫られている事情もある。また、API を IT システムごとに個別対応してきた企業は API が様々なスサノオ・フレームワーク領域の IT システムで利用されるに従い、API を管理・制御していくことが困難になっていることが背景としてあげられる。API の設計仕様定義、セキュリティ設定、トラフィック量コントロールなどを、一

つ一つの API ごとに個別対応していく、制御していくことは多大な時間がかかるだけでなくそもそも管理・制御することが事実上不可能な状態になっていく。そのような管理・制御が難しい状況を放置したままトラブルのない安定した IT システム運用、セキュリティ上問題ない IT システム基盤を実現することは難しい。そのような状況を打開するため、API ゲートウェイ構築や API 管理ツール導入によって、API を統合的・網羅的に管理・制御しようとする動きが出てくることが多い。

◆ API（群）個別

■ 設計思想・方針の有無

少数の API を利用する段階から、統合的に多数の API を活用して DX を推進していく段階に進み API 基盤・API ゲートウェイを構築する際に API 設計仕様を標準化する取り組みが行われている。社内の様々な IT システムに存在している仕様の違う API を使い続ける場合、開発者が API の仕様を理解することに時間がかかる。そのため、API 設計仕様を標準化することで仕様を理解する時間を短縮し開発スピードを向上させることができる。

また、API 設計時にはデータベースのトランザクション確保も考慮されている。API でやり取りされるデータ長が長い場合などで API での処理に時間がかかり処理中に IT システムトラブルが発生した際には、API 処理により更新する複数データベース間でデータ不整合が発生するリスクが高くなる。そのような状況を避けるために一つのトランザクションは一つの API で完結するよう設計しているケースがある。基本的な考え方としては、データストアの読み取りと更新の操作を分離する CQRS（Command and Query Responsibility Segregation）などを実装していくことが重要である。リクエストは、リクエストで受け止めて MQ（メッセージキュー）に一旦いれて、セッションとして終わりにする。MQ に入ったキューがしっかりトランザクションになっていき、エラーが起こったらロールバックするという仕組みを実現することが理想的である。

API や API パラメータの命名規約を定めることが行われている。命名規約として URL に必ずバージョンを入れることや、バックエンドの API なのかロジックを含んだ API なのか必ず S や P など頭文字をつけて、どういうレイヤーの API なのか表現するといった対応が取られているケースがある。そうすることで、API を検索しやすくなり API のメリットである API の再利用が可能になる。

外部サービス提供 API を利用する場合は、外部サービス提供 API が利用不可になるリスクを考慮しておく必要がある。API 提供ベンダーとの SLA 締結時に、利用不可時の対応を、API が使われている IT システムの重要性に応じて交渉する対応を行っているケースがある。

また、API ファーストのサービス構築が行われているケースもある。例えば、EC サイトなどでフロントエンドとバックエンドを分離させ、フロントエンドの画面(UI)の見た目や操作性を重視させバックエンドも API 化して柔軟に情報をやりとりさせるといったプロジェクトが進んでいる。そのような取り組みによってバックエンドの基幹システムの改修を極力発生させず、フロントエンドの EC サイトでの顧客体験を向上させることができる。

■ API の非機能要件担保方法

API の非機能要件を担保するための取り組みとして、API 設計時にパフォーマンス定義を行っている。API を介してデータのやり取りをする送信側 IT システムと受信側 IT システム両方のパフォーマンスを考慮して設計している。例えば、基幹系システムは性能限界があり急激なボリューム増加に対応することが難しく、システムダ

ウンした時には他 IT システムや業務遂行に大きな影響が出てしまう。そのような事態を避けるためにあらかじめどのくらいのデータ量をどのくらいの頻度でやり取りするか、基幹システムにて対応可能なレベルをパフォーマンス定義として事前に定め API 設計を行っている。また、API ゲートウェイを構築した場合は、API ゲートウェイを経由せず個別 API で直接連携していた時のパフォーマンスを基準として、その基準を大幅に下回らない水準をパフォーマンス定義の目安としている。パフォーマンス定義を行う際は API を利用する IT システムの可用性を考慮している。可用性が高い IT システムが利用する API に対してパフォーマンス定義をより厳密に行っている。

API リリース後は、API 管理ツール上で定義したパフォーマンスを閾値として設定して API の負荷制限を行っている。例えば、API 管理ツールが提供するプロダクト機能（API をパッケージ化してアプリケーション開発者に公開する仕組み）が負荷制限などに利用されている。また、API の利用が定義したパフォーマンス水準に近付く、さらに超えていく場合を想定して、API ゲートウェイを業務種類別にサーバーを分けることや、想定パフォーマンスに近くなったらスケールアップやスケールアウトするようにしている。API リリース後、API 項目情報を追加する必要がある際は、項目情報追加がパフォーマンスに影響するため API の性能拡張・増強を行っている。

■ API のセキュリティの設計、運営思想

API のセキュリティを確保するために、管理ツールで API アクセスキーを発行して、API アクセスキーがないとアクセスできない仕組みにしている。アクセスキーを発行する場合、API 開発時のチェックリストをすべて合格しないと発行しないといったワークフローを定めるなど、API の全体管理を行う部署がガバナンスを利かせてアクセスキー発行を管理している。

その他の取り組みとしては、権限の認可や認証などの外部認証を通ることをポリシーとして設定している。また、データの流通経路を意識することも重要である。オンプレミス上に API 管理ツールのモジュールを導入し、オンプレミスの中でクローズしてデータ処理の終わったものだけをクラウドに出すようにすることで、不必要なデータをクラウドに出さないといったケースもある。

海外では API に関するセキュリティ上の問題が発生している。一方、国内ではセキュリティ上の問題は多く発生していない傾向にある。現状、API ゲートウェイだけがパブリックネットワークに出ていて、JSON 攻撃などの防御機能もあるためセキュリティ上の問題があまり起きていない状況である。

■ API の開発要件

IT システム間のつなぎこみを容易にするために、API ゲートウェイを構築し API 管理ツールのコネクタ機能を利用している。コネクタ機能を利用することでデータベースごとのつなぎこみ作業が軽減され、開発スピードを向上させることが可能になる。

また、API 開発において API 管理ツールが提供するノーコード・ローコード開発の機能を活用する事例も見られる。API を一から開発するのではなく既存 API をコピーして開発することで開発生産性が 10%～20%向上するケースもある。さらに開発スピードを向上させつつ、熟練の技術者でなくても API 開発の品質を高く保つことができるケースもある。

API の高い開発品質を確保するための取り組みとして、開発プロセスにおいて満

たすべき開発要件をチェックリストにして確認する取り組みがある。API 開発は外部ベンダーに依頼するケースも多いため、あらかじめ API が満たすべき項目のチェックリストを示すことで、開発者は満たすべき項目を加味して開発を進めることができ、高品質な API 開発が可能になる。また、最初の段階では、満たすべきチェック項目の数を少なくして開発しやすくし、開発段階が進むにつれてチェック項目を増やすといった工夫を行いながら、開発品質と共に開発スピードを向上させている。

■ API を介したデータの利用・活用

API を介したデータは、ビジネス部門が自分たちでデータ分析ツールを利用して分析を行う際のデータ源となっている。API を介したデータは、経営レベルでも使用されているが、どちらかというとビジネス部門側で使用されるケースが多い。

リアルタイムデータ利用に関して、分析の観点からの API を介したリアルタイムデータ利用のケースは、検討されることはあるが、実現例は少ない。実現例としては、新規に API を開発するのではなく、既に存在する API 間の連携をリアルタイムにする事例（社内的な為替レートや外部から取得した気象情報など）がある。現状 5 分に 1 回程度起動するバッチ処理(マイクロバッチ)で対応することでリアルタイム連携に近い効果を実現している事例がほとんどである。

イベントドリブンの API により基幹システムから顧客向け IT システムへリアルタイム連携する取り組みは、一部の企業で行われている。リアルタイムなデータ連携を前提とした業務オペレーションが組まれている場合、イベントドリブンの API 導入の効果は大きい。例えば、EC サイトにおいて店頭で販売している商品在庫を引き当てる仕組みになっているケースがある。受注した場合、すぐに店頭で商品を確保する必要があるが、API でリアルタイム連携することで、店員はすぐに該当商品を EC 販売分として確保することができる。その結果、EC 販売在庫を店頭在庫の他に持つ必要がなく在庫回転率が向上し、店頭だけでなく EC で全国に販売できることで販売機会を増加させることに成功している。

一方、IoT によるイベントドリブンの API 事例は極めて少ない。現状 PoC が少しある程度しかない。POS 端末やハンディ端末などデバイスとの連携においても専用ツールで行うことが多く、API 連携は使用されていないケースが多い。

◆ API 全体管理

■ API の開発要件

API ゲートウェイを構築し、API 管理ツールを導入して API の全体管理を行っている。API 仕様を標準化し管理ツールのコネクタ機能を活用して API 接続しやすいようにすることで、開発スピードを向上させている。また、API プロダクト機能を活用してアクセス権のコントロールやアクセス量レート制限を行っている。API ゲートウェイの API を集約し API 管理ツールを活用することで社内に散在する API を漏れなく統合的に管理することができるようになる。

また、技術的な取り組みに加えて API ゲートウェイを管理する部署が API のガバナンスを行っている。API 設計・開発時の API 開発チェックリストの導入・確認やユースケースによって既に同様の API が存在していないか、開発の優先順位として妥当かといった確認などを行っている。ノーコード・ローコード開発により API が

開発しやすくなるとAPIが濫造されるリスクが高まるため、API開発のガバナンスがより重要になる。APIのガバナンス対策としては開発環境と本番環境を分離することや、人ではなくCI（継続的インテグレーション）ツールを活用して、開発したAPIのデプロイメントを行う取り組みが行われている。また、API管理ツール上でAPI同士の関連図や使用回数を確認して利用の少ないAPIを棚卸し、APIの統合や廃止を行っている。そのような取り組みを行うことで正当なプロセスを経ないAPIが本番環境に存在する状態を防ぐことができ、本番環境へのデプロイメントのログが残り後から調査できるようになる。APIの棚卸によって使われていないAPIを削減して、リソースを節約することが可能になる。

■ APIの運用要件

APIリリース後、安定したITシステム運用のためにAPIの監視を行っている。監視ツールを活用してトラフィック量、応答速度、エラー件数などのAPIの運用状況を確認できるようにしている。確認項目に閾値を設定し閾値を超えた場合にアラートを出すようにしている。監視ツールで発見した問題にすぐ対処するだけでなく、例えば、1週間といった少し長い期間で運用状況に特異点がないかを確認し問題が発生する前に特異点に対する処置を行い、将来的な問題発生リスクを低減する取り組みも行われている。また、APIの運用状況・障害の分析をするだけでなく、サイトの導線など、顧客行動も分析することで、サービス利用増加とAPI利用状況の関係性を分析して新たなサービス開発に活かすといった取り組みも行われている。

APIの障害対応としては、基幹システムやECサイトなどミッションクリティカルなITシステムに検知用のAPIポリシーをいれてシステムトラブル情報を伝え、それ以降のリクエストを遮断する取り組みが行われている。検知用のAPIポリシーをITシステムに設定することで、輻輳が輻輳を呼びITシステム全体がダウンすることを防ぐことにつながる。また、特に顧客向けITシステムなど、APIの利用状況やパフォーマンス状況を、事前に完全な形で想定してテストを実施することは費用対効果の観点で現実的ではない。そのため、事前対策を行うとともに障害やパフォーマンスに関する問題が発生する前提で、問題が起こった場合の初動をいかに早くできるか、関係するITシステムを含めて影響範囲を想定して対処できるか、といった事後対策の準備も行われている。

開発者がAPIを開発しやすいようにするためにAPIのカタログ管理を行っている。自社で社内サイトを構築してAPIカタログ管理を行う事例や、管理ツールを活用する事例がある。自社開発APIをカタログに登録するだけでなく、管理ツールで外部サービスプロバイダーが提供するAPIも表示されるようになっている。

APIのバージョン管理においては、APIゲートウェイを構築してAPI管理ツールを利用する取り組みが行われている。ITシステムごとの個別APIをバージョンアップする場合、APIのバージョンアップに伴うインターフェースを変更する必要があるため、関係するITシステムと連携してバージョンアップを行う必要があり、APIが多数のITシステムに使用されている場合は、影響を受けるITシステムを調査することに時間がかかるだけでなく、変更漏れによるITシステムトラブル発生のリスクが高くなる。一方、APIゲートウェイを構築してAPI管理ツールを導入している場合は、管理ツール上で、APIに接続しているITシステムを漏れなく把握することができるので、影響を受けるITシステムの調査に時間がかからなくなり、バージョンアップ作業がスムーズになる。

API の外部公開について、構想段階のものはあるが、サービスインしているケースは少ない。API を外部公開する際は、管理ツールが外部公開機能を持っているので、ツール内で外部サイトを作って公開することができる。サイト上に API 仕様情報や API のモックを掲載している事例がある。また、API 外部公開で課金する場合は、管理ツールから API 使用回数情報を抜き出し、課金管理用の SaaS ソフトにデータを渡して課金する形になる。

4.2. DX 推進に向けた API 活用の課題と対策

今回API活用の取り組みが進んでいる4社の企業と、API管理ツール提供企業1社にヒアリングを実施した。一方、API をまだ十分に活用できていない企業も多く存在している。そのような企業が、今後 DX 推進に向けた API 活用を進める上での課題と対策を、以下に考察する。

◆ セキュリティ

API に関して、セキュリティは最も重要である。API は自社の IT システム、アプリケーション、サービスに通じる「ドア」である。そこには、リスクとセキュリティの脆弱性に関する可能性が無数にある。新しい API を追加するたびに、攻撃にさらされる可能性が高くなる。実際、毎年 API の悪用によって多くのユーザーの個人情報や機密情報が盗まれている。最も脆弱な API とは、存在を知らない API である。対象を知らなければそのセキュリティを確保することはできない。非公開であるが、Web またはモバイルアプリケーションの一部として使用されている、いわゆる「シャドーAPI」が無防備に放置されていることが多い。したがって、死角を作らないために、API を検出することが重要になる。対策としては、API セキュリティガバナンスポリシーの策定、API 検出・API 不正利用検出を行うための専門ベンダーが提供する API セキュリティソリューションの導入、API デプロイメントでの CI ツール導入、API 使用実績に基づく API の棚卸と整理などがあげられる。

また、API カタログ管理も重要である。使用している内部開発 API、外部サービス提供 API を API カタログとして登録する、または開発者に登録させることが必要である。利用する API の数が増えてきた場合、品質的にどの API が良いのか、わからなくなってくる。データカタログのように、API に関する属性情報を付け、API に関する情報を開発者に流通させていく仕組みが必要である。今回の企業ヒアリングでは、API を開発する際に API のユースケースを確認・承認しないと API を開発できないようにしており、社内向け Web サイトを構築して、開発者向けに API に関する情報を掲載していた。

◆ 非機能要件の担保

API を限定されたデータ量や頻度でインターフェースの一つとして利用している段階では課題にならないが、単一の API を複数のスサノオ・フレームワーク領域や IT システムから利用する段階になると、どのように非機能要件を担保するかが課題になる。特に、顧客向けサービス用に API を利用する場合や API を外部公開する場合、どのくらい API が利用されるか、どのくらいのデータがどのくらいの頻度でやり取りされるかを、正確に予測することは難しい。そのため、たとえ精緻なパフォーマンス定義を行い、API を開発・導入したとしても、API を利用する中で、非機能要件を担保できない状況が発生しうる。

対策としては、API の性能を増減できる仕組みを準備しておくことが必要である。アクセス量が増えた場合に、API ゲートウェイを増やすことや、業務種類別にサーバーを分け負荷分散を行っていた。また、想定パフォーマンスに近づいてきた場合は、スケールアップまたはスケールアウトするようにしていた事例が該当する。

◆　全体管理・ガバナンス

今後、利用するAPIを増やしていく場合に、社内に散在している様々なAPIをどのように管理していくか、ガバナンスをかけていくか、が課題になってくる。全体管理・ガバナンスが十分にできなかった場合、セキュリティの項目で取り上げた「シャドーAPI」を発生させることになる。また、API設計仕様の標準化や全体管理プロセス導入、API開発運用環境整備を推進しなかった場合、開発スピード向上や安定したAPI運用を実現することができず、API活用によるメリットを享受することができなくなる。

APIの全体管理・ガバナンスを強化する対策としては、技術的な側面と人的な側面がある。技術的な側面としては、APIゲートウェイ構築による一元管理やAPI管理ツール導入があげられる。一方、人的な側面としては、APIの全体管理・ガバナンスを行う責任者や組織の設置、APIに関するプロセス・ワークフローの設計・導入、ポリシー作成などがあげられる。今回の企業ヒアリングでは、APIゲートウェイの構築・運用を担う部署が、API全体管理・ガバナンスを担う役割であり、API開発のチェックリストやユースケース確認などのプロセスを導入・運用していた事例が該当する。

◆　API新技術への対応

APIに関する技術は、日々進化しており、新しい技術に対応していくことが課題となる。現在、APIの主流は、Rest APIと言われるRESTと呼ばれる設計原則に従ったAPIである。一方、昨今では大量のデータを丸ごと抽出したい、必要なデータセットだけを取り出したいというニーズから、APIクエリ言語であるGraphQL APIと言われる技術の活用などが増えてきている。これは、SQL構文を投げ回答を受理する方式のため、Rest APIを何回も呼び出すよりも、効率的にデータ連携することが可能である。

APIの新しい技術へ対応するためには、他の技術領域と同様に、普段からAPIの新技術に関する情報収集や調査を行うことが必要である。情報収集・調査した結果を元に、定期的に現在の自社のAPI技術水準を評価することや、API新技術を自社に適用するケースの検討を行うことが必要である。

◆　API活用推進人材

前述のAPI全体管理・ガバナンスを行う責任者や組織を設置しようとした場合、その役割を果たすことができる人材が社内にいるかが課題になる。

少数のAPIを活用している段階から、一元的・網羅的に多数のAPIを活用する段階に移る際には、全社的にITシステム基盤をどのようにしていくかといった構想立案や、APIゲートウェイ構築など具体的なITシステム基盤の設計を行う必要がある。外部ベンダーやITシステム子会社に大きく依存していて、社内にそのような役割を担う人材がいない企業においては、APIに関する統合的な取り組みを進めることが困難である。

また、前述のAPIの技術的な進化に対応していく際には、新技術にアンテナを張って情報収集や調査を行い、自社への新技術適用の可能性を探ることができる人材が必要となる。

さらに、技術的にAPIを活用するということだけでなく、APIを活用してビジネス上意味のある成果を生み出そうとすると、IT部門だけでなくビジネス部門と連携して仕事を進めることになる。様々なビジネス部門と連携できる人材、ビジネスニーズを理解してAPIを含む技術的基盤に落とし込める人材が必要になる。

API活用推進人材不足に対する対策は、IT人材不足やデジタル人材不足に対する一般的な対策と同様に、社内人材育成、外部人材採用、外部サービス提供会社の活用があげられる。

「API の活用に関する実践状況調査」概要報告書

【全体監修】

ＤＸ推進部　境 真良

【報告書編集】

ＤＸ推進部

鎌田 高輝

西本 靖

佐々木 伸一

中川 貴之

佐藤 弥生

【調査協力】

ガートナージャパン株式会社

【専門委員】（敬称略）

名古屋国際工科専門職大学 工科学部 情報工学科 学科長　山本 修一郎

【改版履歴】

令和 4 年 10 月 26 日 第 1 版 発行

【監修所】 独立行政法人情報処理推進機構（IPA）

【発行所】 独立行政法人情報処理推進機構（IPA）社会基盤センター DX 推進部

〒113-6591

東京都文京区本駒込二丁目 28 番 8 号文京グリーンコートセンターオフィス

URL : https://www.ipa.go.jp/