ノーコード・ローコード開発を成功に導くための「ノー&ローコード虎の巻」の提案
Suggestion of ″No & Low-code Toranomaki″ for the success of no-code/low-code development

リーダー：内田　泰介（株式会社日立ソリューションズ・クリエイト）
研 究 員：池田　幸子（株式会社日立システムズ）
　　古里　透　（株式会社アズビル）
主　査　：山田　淳　（株式会社東芝）
副 主 査：田中　桂三（オムロン株式会社）
アドバイザー：中森　博晃（パナソニック コネクト株式会社）

研究概要
ノーコード・ローコード開発ツールを利用したソフトウェア開発は歴史が浅く,プロセスやノウハウが纏まっていない.また「開発未経験者でも可能」「高品質」「高生産性」といった言葉を信じて開発・保守体制構築や作業計画・コスト確保等の準備を十分行わずに開発を進めてしまうと開発中の QCD 面や開発後の保守面でのトラブルが起きる可能性がある.
本研究ではノーコード・ローコード開発を成功に導くため,陥りやすい注意点を「ノー&ローコード虎の巻」として整備し,開発・保守担当者が実際のプロジェクトで適用することを目的とした.
また,「ノー&ローコード虎の巻」の有効性を検証するために,アンケートによる評価を実施した.アンケート結果を分析した結果，プロジェクト開始前に「ノー&ローコード虎の巻」を用いた事前チェックを行うことでプロジェクトのリスクを検知し，トラブル回避に有効であることが確認できた.

1. はじめに
近年,ノーコード・ローコード開発ツールを利用したシステム開発が増え始めているが,従来のプログラミングを伴うシステム開発と開発プロセスが異なることから,従来通りの品質管理・評価手法が適用出来ない可能性がある.
本研究では,ノーコード・ローコード開発の経験・知識を持っている開発現場担当者視点での「ノー&ローコード虎の巻」を作成し，ノーコード・ローコード開発の品質確保やトラブル回避 に役立てることを狙いとした.
以降 2 章では研究の背景を説明し,3 章では研究提案として仮説立案から「ノー&ローコード虎の巻」作成,仮説の検証方法について説明する.4 章では「ノー&ローコード虎の巻」に対する検証結果について説明する.5 章では,「ノー&ローコード虎の巻」の有効性について考察する.最後に 6 章で,研究成果全体のまとめや,今後の展開について述べる.

2. 研究の背景
2.1 ノーコード・ローコード開発の定義
本研究では，ノーコード・ローコード開発について,次の通り定義する.
可能な限りソースコードを書かず,パラメータの設定や GUI 操作により Web サイトやアプリケーションを開発すること.[1]

2.2 ノーコード・ローコード開発の歴史と期待
最初に登場した Word や Excel，Wix などでは，文書作成・関数処理・図形作成・サイト作成が，コードを書かずに実現できた．しかし実際にはできることに限りがあり，ソフトウェア開発としては実用には至らないものであった.

それが，数年前からノーコード・ローコード開発ツールが進化し実現可能な範囲も広がっていることから，ソフトウェア開発の最先端として注目を集めている．さらに，企業の競争力向上を目的として DX 推進が求められており，通常のソフトウェア開発よりも短期間での開発が可能とされるノーコード・ローコード開発ツールに対する需要は，今後一層高まっていくと考えられる．[1]

この状況の中で，ノーコード・ローコード開発に対する過度な期待が生じて，設計・実装スキルがなくとも開発可能であるといったことや，開発期間の大幅な削減が可能であるといった期待感を持たれることがある．事例として，研究員の所属会社ではノーコード・ローコード開発ツールとして「SalesForce」を利用した開発を行っており，製品仕様が業務にマッチしなかったなどの問題が発生している．

2.3　解決すべき課題（研究ゴール）

2.2 章に記載したようなノーコード・ローコード開発に対する過度な期待感をなくし，本来のメリットを生かすために，我々は研究ゴールを以下の通り設定する．

【研究ゴール】

ノーコード・ローコード開発への期待感を鵜呑みにせず，適切に開発・保守を準備できるようにする．

3.　研究提案

3.1　課題達成への仮説

3.1.1　仮説の立案（ノー＆ローコード虎の巻）

2.3 章の研究ゴールを達成するために，我々はノーコード・ローコード開発に対する期待感とその実態を調査・確認し，ギャップがある部分に対して十分考慮し準備しておくべき留意点を設けることで，より安全にノーコード・ローコード開発プロジェクトを成功に導けると仮説を立てた．

そこで，本研究では開発時や保守時の各場面において留意点を纏めた「ノー＆ローコード虎の巻」を作成する．開発保守メンバがプロジェクト開始前に「ノー＆ローコード虎の巻」を参照して，留意点をチェックし対策をすることで，ノーコード・ローコード開発のメリットを生かした開発保守作業が実現できることを期待する．

3.1.2　ノー＆ローコード虎の巻の説明

「ノー＆ローコード虎の巻」はノーコード・ローコード開発における過度の期待感を抑制する為に留意点をまとめたリストである．本リストは「付録 2.ノー＆ローコード虎の巻（アンケート用）」の「ノーコード・ローコード開発の留意点」の列を参照のこと．

#	SWEBOKの 知識領域	ノーコード・ローコード 開発における期待感	ノーコード・ローコード開発の失敗事例	ノーコード・ローコード開発の留意点
1	ソフトウェアの要求	期待感A 業務用の製品であるため、ある程度のことは意識せずとも何でもできる	■開発事例からのフィードバック 帳票添付機能が使える製品であったが、添付件数は１帳票のみで複数帳票は添付することができず、顧客の要望とあわないことがあった	ノーコード開発であっても、実現したい業務フローやユースケースを作成する必要がある。 （テスト実施時にも必要となる）
2	ソフトウェアの要求			従来の開発と同等に製品に合わせたビジネスモデルや業務フローを提案できる人がいること。
3	ソフトウェアの要求		■他研究コースからのヒアリング 顧客とのSLAによって、製品側の問題であってもこちら側の問題になることがあった	利用する製品のSLAがどの範囲となっているか認識しておく必要がある。
4	ソフトウェア設計	期待感B 設計不要 開発初心者でもOK	■他研究コースからのヒアリング お試しで、ローコードで開発を実施したが、データモデルを設計せずに進めたため、汎用性のないものが出来上がってしまった ■データ構造適正化ができておらず、影響分析のワードが定まらないことや、認識齟齬によるトラブルになる事例もあった。	データモデルや構造設計を明確に理解している必要がある。 （例：同じ意味で異なる名称のデータがあると、複雑になる）

図 1．ノー＆ローコード虎の巻（抜粋）

3.1.3　ノー＆ローコード虎の巻の十分性評価

「ノー&ローコード虎の巻」の十分性（研究ゴールを達成するか否か）を評価するために,我々の研究ゴール,研究課題,評価基準として以下の図 1 の通り GQM 手法（Goal Question Metrics）を用いてまとめた.GQM 手法は,ソフトウェアのメトリクスを計測の目標と質問から得るという目標指向の測定法で,計測活動を行う際に広く用いられ,Basili と Weiss により 1984 年に提案された手法である[2].我々は GQM が本研究ゴールに対し,研究仮説を定量的に判断できる手法と判断し,採用する.

本研究における GQM を以下の図 2 に示す.

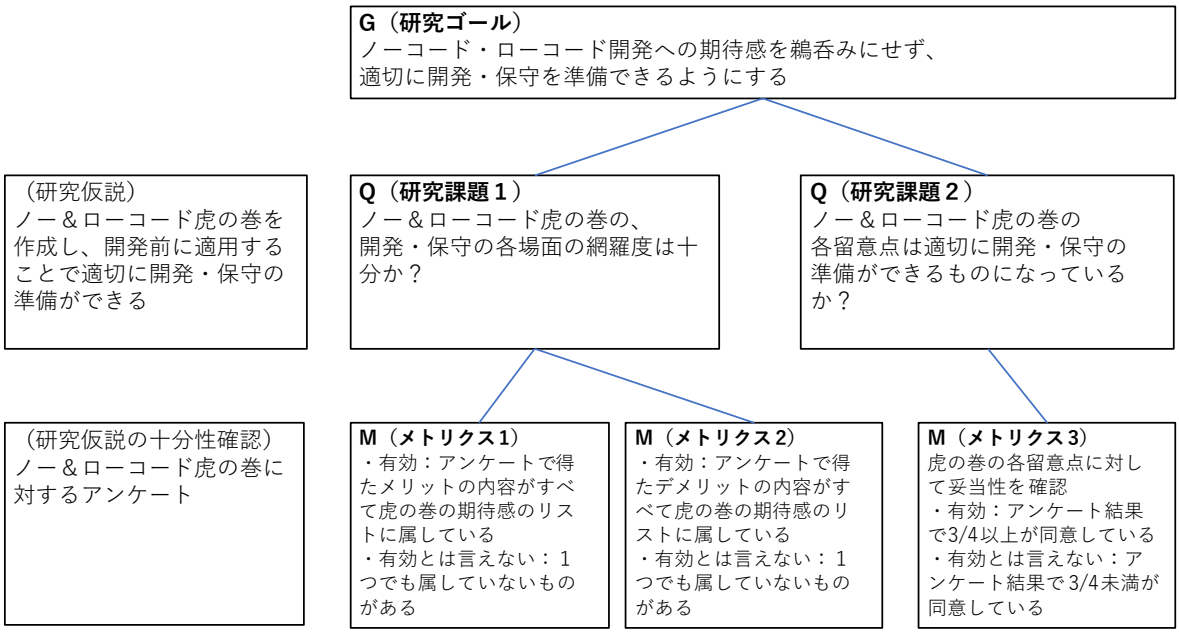

図 2.GQM 手法による研究ゴール,課題,メトリクス（十分性確認）の関係

研究課題 1 を設定した理由は,ノーコード・ローコード開発のメリットを十分生かすうえで,開発保守場面の網羅度（量的な側面）を確認することが必要であると考えたためである.本課題の有効・無効の評価としてメトリクス 1,2 を設定した.

また研究課題 2 を設定した理由は,虎の巻の留意点の内容が開発保守の現場で有効か否か（質的な側面）を確認することが必要であると考えたためである. 本課題の有効・無効の評価としてメトリクス 3 を設定した.

3.1.4 ノー&ローコード虎の巻の作成手順

ノーコード・ローコード開発の留意点のリストについては以下の手順で作成した.

はじめに, 留意点を抽出するにあたり,体系的・網羅的に開発保守の各場面を押さえるべきと考えた.そこで,我々は「software engineering body of knowledge（ソフトウェアエンジニアリング知識体系）」（以降,SWEBOK）[3]を採用した.なぜならば SWEBOK は IEEE が纏めたソフトウェアエンジニアリングに関する知識体系であり,開発保守を体系的に整理しているため,網羅的な抽出に適切と考えたためである.

次に,留意点の元データとして,SWEBOK の各知識領域「付録 2」の「SWEBOK の知識領域」の列をもとに「ノーコード・ローコード開発における期待感」や「ノーコード・ローコード開発の失敗事例」の列の項目を研究員が所属する組織の事例や,他の研究会へのヒアリング,書籍[1]などから収集した.

最後に,「ノーコード・ローコード開発の失敗事例」と「ノーコード・ローコード開発における期待感」の乖離を明確にして,それらを「ノーコード・ローコード開発の留意点」

の列に定義した.

3.2 仮説の有効性の評価方法

「ノー&ローコード虎の巻」の有効性を検証する方法を検討した.本研究の期間内で「ノー&ローコード虎の巻」を適用出来る実際の開発案件を選定することは困難だったため,「ノー&ローコード虎の巻」についてアンケート(詳細は「付録 1.アンケート」を参照）を取り，「ノー&ローコード虎の巻」の各留意点を参照することとした．アンケートの対象者は,ノーコード・ローコード開発の開発や品質管理の経験がある人物を想定とした.

「ノー&ローコード虎の巻」の評価判断基準は図 1 の GQM のメトリクスに基づき以下の 3 点とした.

(1) メトリクス 1：アンケートで得たメリットの内容がすべて虎の巻の期待感のリストに属している場合「有効」と判断する.

(2) メトリクス 2：アンケートで得たデメリットの内容がすべて虎の巻の留意点のリストに属している場合「有効」と判断する.

(3) メトリクス 3:虎の巻の留意点に対して，5 段階(「1．その通りだと思う」「2．一部当てはまると思う」「3．あまりそうは思わない」「4．まったく当てはまらない」「5．意図が分からない」）で評価し,アンケート集計結果を 10 点満点に換算し,7.5 点以上(3/4 以上)の点数を被験者の同意を得たものと考え,「有効」と判断する.

4. 実験結果

仮説検証では,ローコード・ノーコード開発の担当者と管理者や今後導入を検討している組織の開発者・品証部門,計 31 人を対象にアンケートを実施し，ノーコード・ローコード開発のメリット・デメリットの確認と,「ノー&ローコード虎の巻」の効果有無を点数化して確認した.被験者については,「付録 6.被験者のプロファイル」を参照のこと.

4.1 メトリクス 1 に対する結果

メトリクス 1 に対しては有効であるという結果となった.それはアンケートの「(7-1)メリット」で選択された全ての項目がノー&ローコード虎の巻の C 列「ノーコード・ローコード開発における期待感」を選択していることを確認できたためである（選択率 100%）.これにより,開発保守場面における虎の巻の期待感の網羅性は十分であると判断した.結果は表 1.および「付録 3.アンケート（メリット）集計結果」を参照のこと.

表 1．ノー&ローコード虎の巻のアンケート結果（メリット）

開発時のメリット	件数	虎の巻の選択比率
【期待感 A】業務用の製品であり,ある程度のことは何でもできる	2	100%
【期待感 B】設計不要.開発初心者でも OK	10	
【期待感 C】開発初心者でもすぐに利用可能	9	
【期待感 D】カスタマイズや独自仕様の取り込みも容易に行える	2	
【期待感 E】テストはすべて自動で実行ケースも作成しないで OK	1	
【期待感 F】自動生成部分の品質は担保済みであり,テスト不要	5	
【期待感 G】保守不要	0	
【期待感 H】いつでも,手軽に製品も使い続けることができる	0	
【期待感 I】製品側で構成管理も含めては管理してくれる為,不要	0	
【期待感 J】海外製品も自由に利用できる	0	
【期待感 K】個人情報の管理も楽になる	0	
【期待感 L】海外製品も無料で利用可能,コストを抑えることが可能	1	
【期待感 M】顧客要望をすぐに取り込める為,要求仕様の適合は高い	1	
上記以外の期待感を選択した件数	0	0%
計	31	100%

4.2 メトリクス 2 に対する結果

メトリクス 2 に対しては「有効であるとはいえない」という結果となった.それは，アンケートの「(7-2)デメリット」の選択肢にない項目が 2 件あり選択率が 100%ではなかった為である.一方でその他の 29 件については虎の巻の D 列「ノーコード・ローコード開発の失敗事例」と E 列「ノーコード・ローコード開発の留意点」に紐づいていることを確認でき,概ね網羅的に留意点は取り込めていると考える.結果は表 2.および「付録 4.アンケート（デメリット）集計結果」を参照のこと.

表 2. ノー&ローコード虎の巻のアンケート結果（デメリット）

ノーコード・ローコード留意点	件数	虎の巻の選択比率
【留意点 A】実現したい業務フローやユースケースを作成する必要がある	10	94%
【留意点 B】製品に合わせたビジネスモデルや業務フローを提案できる人がいること	2	
【留意点 C】利用する製品の SLA の範囲を認識しておく必要がある	0	
【留意点 D】データモデルや構造設計を明確に理解している必要がある	0	
【留意点 E】担当者の製品理解度はバラツキが出てきてしまうため,設計力のあるメンバを配置する必要がある	5	
【留意点 F】標準的な機能のみで要件は満たせるか確認する必要がある	10	
【留意点 G】すべてのテストが自動で実施されるものではなく,単体,結合テストは実施する必要がある	0	
【留意点 H】ブラックボックスのテストまでしかできない為,保守に備え,ログ出力や調査要求などの契約が必要	0	
【留意点 I】互換性がなくなった場合のリスク対策が必要	0	
【留意点 J】製品の VersionUp 時に発生する影響の見極めが必要	0	
【留意点 K】販売実績のない製品や販売終了後のサポート有無が不明な場合,製品が使えなくなるリスクを踏まえた上で採用する	0	
【留意点 L】複数人で開発する場合は構成管理を実施する必要がある	0	
【留意点 M】海外の製品を利用する場合は,使用許諾を十分理解し,許諾を遵守する必要がある	0	
【留意点 N】海外の個人情報については,各国の法令を遵守し,現地のサーバーで管理する	0	
【留意点 O】有料プランを利用する際は,逆にコストが高くなる場合がある.また,自由度がない分,逆に改造の手間がかかりコストが高くなる可能性がある	2	
【留意点 P】高度あるいは特別な要求・品質確保に 100%対応できるものではなく,知識・期間の範囲内で,要求対応・品質確保を行うことを前提にしていることを,ステークホルダーと合意できている	0	
その他	2	6%
計	31	100%

4.3 メトリクス 3 に対する結果

メトリクス 3 に対して 16 項目中 13 項目は「有効」（基準値 7.5 点以上）,3 項目は「有効であるとは言えない」（基準値 7.5 点未満）という結果となった.

結果については表 3.および「付録 5.虎の巻集計結果」を参照のこと.

表 3. ノー＆ローコード虎の巻の留意点のアンケート結果

#	ノーコード・ローコード開発の留意点	1 10点	2 5点	3 1点	4 0点	5 0点	点数
1	ノーコード開発であっても、実現したい業務フローや ユースケースを作成する必要がある。	27	4	0	0	0	9.4
2	従来の開発と同等に製品に合わせたビジネスモデルや 業務フローを提案できる人がいること。	23	8	0	0	0	8.7
3	利用する製品のSLAがどの範囲となっているか認識しておく必要がある。	22	7	0	0	1	8.2
4	データモデルや構造設計を明確に理解している必要がある。 （例：同じ意味で異なる名称のデータがあると、複雑になる）	17	11	2	0	0	7.3
5	担当者によって製品理解度はバラツキが出てきてしまうため、 それなりの設計力のあるメンバを配置する必要がある。	20	5	3	0	3	7.4
6	パッケージに標準的に備わっている機能のみで要件は満たせるか否かを確認する必要がある。 （満たせていない場合は、それなりの要員を配置する必要がある）	23	6	1	0	1	8.4
7	すべてのテストが自動で実施されるものではなく、 単体、結合テストは実施する必要がある。	22	3	2	0	4	7.6
8	ブラックボックスのテストまでしかできない為、 障害発生する恐れがある。よって保守に備えて、ログ出力や調査要求などの契約が必要。	20	8	0	0	3	7.7
9	互換性がなくなった場合のリスク対策が必要。	23	6	0	0	1	8.4
10	製品のVersionUp時に発生する影響の見極めが必要。	23	6	0	0	1	8.4
11	販売実績のない製品や販売終了後のサポート有無が不明な場合、製品が使えなくなるリスクを踏 まえた上で採用している。もしくはそのようなソフトは選定の対象外としている。	21	6	2	0	1	7.8
12	複数人で開発する場合は構成管理を実施する必要がある。	22	8	0	0	1	8.4
13	海外の製品を利用する場合は、使用許諾を十分理解し、 許諾を遵守する必要がある。	22	6	0	0	2	8.1
14	海外の個人情報については、各国の法令を遵守し、 現地のサーバーで管理する。	20	5	2	0	3	7.3
15	有料プランを利用する際は、逆にコストが高くなる場合がある。 また、自由度がない分、逆に改造の手間がかかりコストが高くなる可能性がある。	20	8	0	0	3	7.7
16	高度あるいは特別な要求・品質確保に100%対応できるものではなく、知識・期間の範囲内で、 要求対応・品質確保を行うことを前提にしていることを、ステークホルダーと合意できている。	21	9	1	0	0	8.3

5. 考察

アンケートの数値結果だけでは分からない定性的な評価を行うため，アンケート回答者の意見を元に，留意点に対しての理解度やアンケート数値の裏付けについて考察した．なお，本考察を踏まえて「付録 7. 改訂版ノー＆ローコード虎の巻」を作成した．

5.1 メトリクス 1 に対する考察

「ノー＆ローコード虎の巻」がノーコード・ローコード開発の期待感の網羅性について「有効」となった理由は，SWEBOK の知識領域を軸に，研究員が所属する組織の事例や，他の研究会へのヒアリング，書籍から期待感を十分に抽出することができたためと考える．

特に SWEBOK の選定については PMBOK，SQuBOK，ISO/IEC25000 など様々な知識体系がある中で設計・実装に関する観点を重点的に押さえることが必要だと考えて採用したことが有効に繋がったと考える．

また，アンケートの結果，選択比率の最も高い項目「開発期間の削減」（39%）について被験者の期待感を考察する．

この「開発期間の削減」について，コメント内容から確認したところ，数的内訳として「設計・実装の削減」9 件，「プロセスの削減」1 件，「要件定義の期間短縮」1 件であった（「付録 3. アンケート（メリット）集計結果」参照）．これより，「設計・実装の削減」について，より踏み込んだ留意点（例：プロジェクトの大小や特徴，設計実装の各 WBS，開発プラットフォームやツールごとの留意点）を充実させることで，より虎の巻を有効活用

できると考える.

5.2 メトリクス 2 に対する考察

ノー&ローコード虎の巻がノーコード・ローコード開発の留意点の網羅性について 16 件中 2 件が留意点のリストに属さず「有効でない」となった理由について考察する.2 件のコメント・考察は以下の通り.

#1「上流工程（業務要件やシステム化要件）が理解出来ていないとノーコード・ローコードでも意味がないと思われる」

これは,実際には留意点には「従来の開発と同等に製品に合わせたビジネスモデルや業務フローを提案できる人がいること」と記載していたものの,留意点のリストから選択されていなかった.これは,留意点を十分理解してもらえなかった可能性がある.留意点には,前提条件や利用場面の具体例追記など,もっと分かり易い記載にする必要があった.

#2「開発費削減の恩恵は誰が受けるのか？原価低減した分がそのまま販売価格低減になったら開発側にメリットは無い.開発に向くが,余程ルールをしっかりしないと稼働後保守には向かない」

#2 については,2 つの問題を指摘していると考える.1 つ目はノーコード・ローコード開発の効率が上がったとしても,利益にならないということである.対策として,ノーコード・ローコード開発の見積については,工数以外（利用者への提供価値など）で示すなどの検討が必要と考える.2 つ目は開発効率化しても保守作業にしわ寄せがいくといったもの.これについては,開発だけでなくプロダクト全体の効率を上げるために,Dev/Ops の概念などで,設計者と保守要員が歩み寄り,保守作業の効率化を検討する必要がある.

アンケートの結果,デメリットと感じているものが最も多かったものは「製品仕様が業務にマッチしなかった」（45%）であった（「付録 4.アンケート（デメリット）集計結果」参照）.これに紐づいたチェック項目が最も多かったものは「#1 ノーコード開発であっても,実現したい業務フローやユースケースを作成する必要がある」であった.これは,製品仕様が業務にマッチしない要件レベルの実現可否を見極める為の留意点であったが,実際は製品仕様にマッチさせるための技術的な設計・実装面での留意点を記載してほしいとの意見もあった.一方で,各社のローカルルールが問題である可能性があり,留意点として汎用的かつ効率的な業務フローに見直すことも含めるべきであると考える.

5.3 メトリクス 3 に対する考察

アンケートの結果,16 件中 13 件が「有効」であったため,ノーコード・ローコード開発における留意点としては概ね活用できるものと考える.このうち点数が最も高い項目は#1 の「ノーコード開発であっても,実現したい業務フローやユースケースを作成する必要がある」で,9.4 点だった.「ノーコードはあくまで「具現化するのに従来よりも簡単であること」がメリットであり,そこに緻密な整合性を取らねばならないような要件が有る限りは,その業務要件調整・整備は無くならない」という意見があった.これは,最大限にメリットを得るためには,製品仕様に業務要件を合わせる調整が必要であり,業務要件と合わない場合は立ち止まって冷静に判断する必要があるものであり,有効なチェック項目であったと判断できる.

一方で,点数が基準値 7.5 点を満たさない(「効果あり」とは言えない）3 項目について理由を以下に挙げる.

留意点#4 「データモデルや構造設計を明確に理解している必要がある」について,「小規模な開発時には当てはまらない」の意見が複数あったもの.これは,小規模の開発であれば設計を意識しないでも,提供されている機能で開発可能というツールの良さの部分について言及されたものであり,大規模開発や複雑な開発では前提条件を補足することが必要であったものと考える.

留意点#5「担当者によって製品理解度のバラつきが出てしまうため,それなりの設計力のあるメンバを配置する必要がある」について,「設計知識よりも製品知識の方が重要」

という意見があったもの.これは,設計力があったとしても製品の理解がない場合,設計不良の問題を引き起こした事例があり,製品仕様についても留意点を補記が必要と考える.

留意点#14「海外の個人情報については,各国の法令を遵守し,現地のサーバーで管理する」について,「個人情報と法令の関連が分からない」という意見があり,理解しづらい留意点となっていた.また「日本か海外かデータの置き場所が重要」といった意見もあり,海外製品を利用する際の法令と,データの置き場所についても追記が必要と判断する.

6. まとめ

6.1 研究成果まとめ

本研究では, ノーコード・ローコード開発への期待感を鵜呑みにせず,適切に開発・保守を準備できるようにすることを目的とした.

これに対して,実験結果,考察により「ノー&ローコード虎の巻」が開発保守の留意点を網羅的（量的）かつ内容的（質的）に概ね満たしており,開発前に適用することで適切に開発・保守の準備ができることを確認できた.理由として, SWEBOK の知識領域を軸に,研究員が所属する組織の事例や,他の研究会へのヒアリング,書籍などから期待感を十分に抽出することができたことである.とりわけローコード開発の留意点として,10 点中 9.4 点の高得点を得た「ノーコード開発であっても,実現したい業務フローやユースケースを作成する必要がある」など,16 項目中 13 項目が基準値以上の評価を得ており,ノーコード・ローコード開発の有効な留意点として十分な成果があると考える.

一方で, 被験者が考えるデメリットの一部が留意点として網羅的に挙げられなかったことや, ノーコード・ローコード開発ツールの採用で開発効率が向上したとしても,そのまま利益に繋がる訳では無い点や,保守要員にしわ寄せが発生するというデメリットに対する留意点が挙げられなかったことが課題である.今後はノーコード・ローコード開発の見積については,工数以外（利用者への提供価値など）で示すなどの検討や,Dev/Ops の概念により設計者と保守が連携して開発工程全体のライフサイクルを検討する必要がある.

今後システム開発はよりスピード感を持った対応が必要になっていき,ノーコード・ローコード開発はその有効な手段の１つである.ノー&ローコード虎の巻を活用することで,短期間かつ高品質なシステム開発の実行を確実に支援していくことが期待できる.

6.2 今後の展開

6.1 章で挙げたように,以下の課題が残っており,今後も「ノー&ローコード虎の巻」のブラッシュアップが必要であると考える.

(1)虎の巻の内容の誤解により,被験者が考えるデメリットの一部が留意点に含まれていないと判断されたため,留意点の誤解防止につなげるよう前提条件を補記するなど記載内容を工夫する.

(2)ノーコード・ローコード開発のメリットから生じる,新たなデメリット（保守工程のシワ寄せ,利益獲得につながらない）に対しては,工数以外（利用者への提供価値など）で示すなどの検討や,Dev/Ops の概念により設計者と保守が連携して開発工程全体のライフサイクルを検討し,虎の巻の留意点に含めていく.

また,仮説とアンケートで立証されたものの,実際の開発保守現場での効果は確認できていないため,今後各研究員の職場で利用し,実効果の検証と新たな留意点を収集し更新していくことで「ノー&ローコード虎の巻」の完成度を上げていく.

参考文献

[1] 森岡 修一, 基礎から学ぶ ノーコード開発, C&R 研究所,2021/3/17
[2] 木村 初夫,CMMI 導入の為の GQM 手法による測定データの研究,SQiP 研究会,2004/
[3] ソフトウェアエンジニアリング基礎知識体系 ―SWEBOK V3.0,オーム社,2014/11/25