第38年度（2022年度）ソフトウェア品質管理研究会（研究コース1「ソフトウェアプロセス評価・改善」）
SQiP研究会アンケート（ノーコード・ローコード開発に関して）　「所要時間15分程度」

【はじめに】

SQiP研究会(日本科学技術連盟の研究会/ 研究コース1「ソフトウェアプロセス評価・改善」）で、ノーコード・ローコード開発の品質管理について研究を行っています。
仮設立証に向けたアンケートにご協力をお願いします。
記載頂いた内容について、研究目的以外で利用することはありません。

【研究のきっかけ・目的】

ノーコード・ローコード開発の製品を利用した開発は歴史が浅く、プロセスやノウハウが纏まっていません。
「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見切り発車で開発を進めてしまうと、トラブルが起きる可能性があります。
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「虎の巻」として整備することを目的としています。

【アンケートの利用目的】

ノーコード・ローコード開発の関係者（開発担当、品質保証担当等）に、「虎の巻」の有効性（研究の仮説）についてアンケートを行い、
回答結果を基に仮説検証を行います。

【本研究におけるノーコード・ローコード開発の定義】

専用の製品を用いてソースコードを書かない、または少ないソースコードでアプリケーション開発を行うことを指します。
（製品の例：「kintone」、「salesforce」）

【アンケート】

1. 会社名と名前を教えて下さい。（記載内容について、後日追加でヒアリングさせて頂く場合があります）

会社名
名前

2. 直近で参画しているプロジェクトにおける役割を教えてください。（当てはまるものを全て選択して下さい）

①プロジェクトマネージャ
②チームリーダ
③開発担当
④テスト担当
⑤品質保証
⑥その他　自由記載欄

3. 経験のある開発プロセスを教えて下さい。（当てはまるものを全て選択して下さい）

①ウォーターフォール型開発
②アジャイル型開発
③その他　自由記載欄

4. ノーコード・ローコード開発の経験について教えて下さい。（当てはまるものを選択して下さい）

①開発、品質管理の経験がある
②経験が無い

5. 4.でノーコード・ローコード開発の経験があると回答した方に質問します。
プロジェクトの概要について、差し支えない範囲で教えて下さい。（①は自由記入。②～④はプルダウンから選択し「その他」を選んだ方は自由記載欄に記入して下さい。）

①製品名
②業種　自由記載欄
③誰が主体的に導入を決めたか　自由記載欄
④導入の1番の目的　自由記載欄

6. 4.でノーコード・ローコード開発の経験があると回答した方に質問します。
開発途中や本番稼働後に発生したトラブルについて、差し支えない範囲で教えて下さい。

7. ノーコード・ローコード開発について、ベンダ側/お客様側目線でのメリット/デメリットについて教えて下さい。
（※4.でノーコード・ローコード開発の経験が<u>無いと回答した方も</u>、想像の範囲で回答をお願いします。）
（プルダウンから1番当てはまるものを選択し、具体的な内容を下の自由記載欄に記入して下さい。）
(7-1) メリット（導入目的が達成できたか）

(7-2) デメリット

8. 「虎の巻」シートを参照し、回答をお願いします。
（※4.でノーコード・ローコード開発の経験が<u>無いと回答した方も</u>、想像の範囲で回答をお願いします。）
SWEBokの知識領域を基に、ノーコード・ローコード開発で陥りがちな「先入観」「事例」と「仮説・チェックポイント」を整理しました。
各観点について妥当と思うか、選択肢から回答をお願いします。
また選択肢3～5を選んだ方で、他により適切なチェックポイントを挙げられる場合は自由記載欄に記入をお願いします。

ノー＆ローコード虎の巻

ノーコード・ローコード開発を行うにあたり、各領域で押さえるべき留意点を挙げました。
これらについて、妥当だと思うか回答下さい。（回答は以下から選択をして下さい）
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

#	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	回答	自由記載欄：3~5を選んだ方は他により良い留意点を記載して下さい。
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	留意点A 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			留意点B 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		■他研究コースからフィードバック 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	留意点C 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	■他研究コースからフィードバック 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 ■他研究コースからフィードバック 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。	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 （例：同じ意味で異なる名称のデータがあると、複雑になる）		
5	ソフトウェア構築	期待感C 開発初心者でもすぐに利用可能	■開発時の欠陥からフィードバック 製品仕様の理解不足により、設計不良を発生させた	留意点E 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		
6	ソフトウェア構築	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	■他研究コースからフィードバック パッケージ製品に対して、独自の改修を行うと影響があり、一定の割合を超えると、通常のウォーターフォール開発よりも生産性・品質は悪化する。 ■他研究コースからフィードバック 製品の機能を利用するよりも、開発者が直接コードを書いた方が自由度が高く作れると判断して、改修量などはあまり考えずに追加機能を実装する事例があった	留意点F 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）		
7	ソフトウェアテスティング	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	■ローコード、ノーコード開発者からのフィードバック 実際の現場バグ一覧を確認した結果、単体レベル、結合レベルの欠陥が摘出されていた。	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。		

8	ソフトウェアテスティング	期待感F ローコード、ノーコード製品が自動生成した部分の品質は担保されているため、テストや保守が不要	■他研究コースからフィードバック 障害発生時にサービス提供ベンダにログ提供を依頼したものの、もらえなかったので、自分たちでログの取得を行った。 ■他研究コースからフィードバック ローコード開発で作られた、システムにおいて、割り込み制御に関する障害が発生。 根本原因の分析を実施したが、仕様がBlackBoxであたため、根本原因にはたどり着くことができなかった。	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、 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。		
9	ソフトウェア保守	期待感G 保守不要	■過去担当PJの経験からフィードバック 製品のVerUpについて 追加で実装したJava Scriptで開発した部分を全て作り直す事が必要があった。データをオリジナル形式で保有しているで、VerUp作業には過去データ全てをコンバージョンする必要がある。	留意点I 互換性がなくなった場合のリスク対策が必要。		
10	ソフトウェア保守			留意点J 製品のVersionUp時に発生する影響の見極めが必要。		
11	ソフトウェア保守	期待感H いつでも、手軽にどの製品も使い続けることができる	■研究員の知見からフィードバック 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	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。		
12	ソフトウェア構成管理	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	■研究員の知見からフィードバック 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。	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。		
13	ソフトウェアエンジニアリング・マネジメント	期待感J 海外製品も自由に利用できる	■他研究コースからフィードバック ノーコード開発をしているアプリが海外製品の場合、個人情報保護の問題、欧州の規格の関係で売却する際に問題になった。	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。		
14		期待感K 開発も簡単になるので、付随して個人情報の管理も楽になる。	■他研究コースからフィードバック GDPR。欧州外に持ち出すのが法に触れる。	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。		
15	ソフトウェアエンジニアリング・マネジメント	期待感L 海外製品も無料で利用可能であったり、コストを抑えることが可能	■書籍からの引用 本格運用となると有料プランを使用しなければならない傾向がある。 ・独自ドメイン ・SEO対策 ・その他、運営に関する詳細設定	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。		
-	ソフトウェアエンジニアリング・プロセス	ー	ー	ー	ー	ー
-	ソフトウェアエンジニアリングのためのツールおよび手法	ー	ー	ー	ー	ー
16	ソフトウェア品質	期待感M 顧客の要望をすぐに取り込める為、要求仕様に対する適合は高いものが作れる	■過去担当PJの経験からフィードバック ユースケースを読み込ませて作成する製品について、ユースケースには業務と適合する為の詳細な記載が無かったが、ベンダ任せとなっていた為、細かい部分の開発はどのように実施しているのか不明のまま開発を進めた。 結果、エラーが多発するなど、開発トラブルとなった。	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。		

#	開発時のメリット		割合	理由	虎の巻の「ノーコード・ローコード開発における期待感」の内容
	メリットの分類	工程（最も割合の大きい開発期間の削減に注目）			
1	開発期間の削減	設計・実装作業		要求仕様が製品の標準機能内で実現可能な場合は、スクラッチ開発部分が減り、開発期間、品質面で有効。	期待感A
2	開発期間の削減	設計・実装作業		ローコードツールを利用した場合、物理設計・実装・単体テストにおける期間短縮に効果があると考えている。	期待感E
3	開発期間の削減	設計・実装作業		プレームワークや部品が用意されているので、設計・実装期間が大幅に削減されることを期待。 運用コストが低い開発環境であれば、トータルの開発・運用コストも抑えられる。	期待感B
4	開発期間の削減	設計・実装作業		既存のパッケージを使用することで開発期間は短縮できると思います	期待感A
5	開発期間の削減	設計・実装作業		製造期間（B2～CT）を短縮できる。	期待感C
6	開発期間の削減	設計・実装作業		いわゆる「ソフトウェアの設計プロセス」としてはなくならず、開発期間が短縮されるイメージです。 プロトタイプ(ないしはそれに近しいもの)は従来よりもある程度短く作れるでしょうから、そこに係る期間は「通常であれば」減ると思われます。	期待感C
7	開発期間の削減	プロセス	39%	・開発環境の仕様の制約があることで、その中での実現機能として着地させることができ、開発期間の短縮ができた。 ・仕様書を起こさなくても、プロトタイピングで進められることで、要求仕様書作成の期間の短縮ができた。	期待感B
8	開発期間の削減	設計・実装作業		・あらかじめ用意されている部品を組み合わせてアプリケーションを開発できるため、開発工程が簡略化され、開発工数を削減できた	期待感C
9	開発期間の削減	要件定義		制約事項になるが、ノーコード開発製品の仕様に依存する部分があり、動作条件を絞ることができた。そのため要求仕様、要件定義で発注者と開発者との合意形成の時間を短時間に抑えることができ、開発期間を短縮できた。	期待感M
10	開発期間の削減	設計・実装作業		標準機能をそのまま使える、もしくは多少のカスタマイズ程度で使用できる分には開発規模自体は少なくなり、開発期間は抑えられる傾向にはある。ただ、実現可能かの精査や問題がある場合の対策にやはり時間はかかる。	期待感D
11	開発期間の削減	設計・実装作業		標準機能で実現できる開発内容で、かつ、製品仕様を理解している人が開発を行う場合、短期間で開発を行うことができた。	期待感C
12	開発期間の削減	–		–	期待感D
13	開発初心者（ユーザ自身）でも開発出来る			想像の範囲で回答します。・初期参入が出来れば競争相手も少ないと思われ、単価も有利な額になっていると想定される。	期待感C
14	開発初心者（ユーザ自身）でも開発出来る			難しい操作も高度な知識も必要ない為、開発が行いやすい	期待感B
15	開発初心者（ユーザ自身）でも開発出来る			業務要件が製品の機能の中で実現できる場合は、開発期間の短縮や開発費用も抑制できる上、開発初心者でも高品質なソフトウェアが開発できる。	期待感B
16	開発初心者（ユーザ自身）でも開発出来る			プログラム言語知識が無くても開発できるため	期待感C
17	開発初心者（ユーザ自身）でも開発出来る			・シンプルなアプリやシステム開発には有効・複数のアプリを連携させることができる	期待感B
18	開発初心者（ユーザ自身）でも開発出来る		32%	・プログラミングに精通していなくても実装が可能である ・採用したシステム次第で一定水準の品質を担保可能 ・実装者による属人化の回避	期待感F
19	開発初心者（ユーザ自身）でも開発出来る			自分達の業務を、より効果的かつ効率的にしていくにあたり、必ずしも専門職でなくとも開発できるということを、ノーコード・ローコード開発を通じてその障壁を越える切っ掛けを得やすい。	期待感B
20	開発初心者（ユーザ自身）でも開発出来る			製品内で提供されている機能（テンプレート）を利用するだけで、簡単なシステムを作成することができる。プログラミングの知識が不要で、GUIで操作可能。	期待感B
21	開発初心者（ユーザ自身）でも開発出来る				期待感B
22	開発初心者（ユーザ自身）でも開発出来る			ノーコード・ローコードの概念が、システム開発知識が乏しくても開発ができると言うことだと理解していますので、 当該選択肢が当てはまるかと思います。	期待感B
23	開発費用の削減			–	期待感L
24	開発費用の削減			開発工数自体は削減できた。	期待感B
25	開発費用の削減		16%	お客様側目線 ・開発コストを抑えることができる。　※ただし製品利用無し開発のコストが不明なため比較できず ・導入実績があることである程度安心感をもって開発に臨める。 ベンダ目線 ・スタート時点で製品有識者を育てておく必要が無い。開発を進めながら有識者教育が可能。	期待感C
26	開発費用の削減			一定の開発費削減は実現できたが、当初ほどの削減効果は出なかった	期待感C
27	開発費用の削減			同じチームのメンバがノーコードツール（Unqork）の研究・支援を実施しているが、他部門からの要望のほとんどが期間の短縮とコストの削減。新規開発でＵＩにこだわらなければコードを書いてのスクラッチ開発と比較すると環境準備や導入教育も軽い。	期待感C
28	高品質			製造不良の作り込みは無さそうなので、品質は良くなるのかと思います	期待感F
29	高品質		13%	–	期待感F
30	高品質			–	期待感F
31	高品質			コーディングミスが減ることにより品質は向上しそう	期待感F

付録4.アンケート（デメリット）集計結果

#	開発時のデメリット	割合	理由	「ノーコード・ローコード開発の留意点」の内容
1	製品仕様が業務にマッチしなかった		ノーコードはあらかじめ用意されている機能のみを使うイメージなので汎用性・機能拡張が容易にできないものと思われ、限定的で大規模システムには向かないように思われる。	留意点A
2	製品仕様が業務にマッチしなかった		全て自由に開発できるわけではないため、ある程度業務をツール側に寄せるようなことが発生する。 それが許されない場合はツールを採用するべきではないが、Fit&Gapを事前検証する時間が必要となる。	留意点B
3	製品仕様が業務にマッチしなかった		プレームワークや部品が汎用的に作られており、自由にカスタマイズすることを想定していないため、運用する組織（会社）独自の仕様には対応が難しいケースがある。	留意点B
4	製品仕様が業務にマッチしなかった		標準的な機能ではできることの範囲が決まっていそう。（想像ですが）	留意点A
5	製品仕様が業務にマッチしなかった		製品が提供する機能は限られているので、業務にマッチさせられない部分もでてくるのでは無いかと思います	留意点A
6	製品仕様が業務にマッチしなかった		但し、「作る期間」が減る可能性は有ると思いますが、そこに係る調整が減るかというと疑問です。結局製品仕様が業務に完全にマッチすることは絶対に無いと思います。それをユーザ側が諦められるか(もしくは代替案を許容できるか)が非常に大きいと思っています。	留意点A
7	製品仕様が業務にマッチしなかった		・自由度が低く柔軟性に欠ける。それにより、本来求める動作を実現できないリスクが生じる ・長期、継続的に利用可能かが採用したシステムにゆだねられることになる	留意点A
8	製品仕様が業務にマッチしなかった	45%	ノーコードプラットフォームの制約により、顧客要求のすべてに対応できなかった。	留意点A
9	製品仕様が業務にマッチしなかった		製品によって出来ることが変わると思われる	留意点A
10	製品仕様が業務にマッチしなかった		業務要件に製品仕様が適合しないことは多々ある。	留意点A
11	製品仕様が業務にマッチしなかった		開発環境の仕様の制約により、当初イメージしたようなＵＩや使い勝手は実現できなかった。 実現できた機能を踏まえて、運用の整理を行う必要があった。（ERPの導入に近いかも）	留意点A
12	製品仕様が業務にマッチしなかった		・従来の開発に比べるとどうしてもできることが限られる点 ・ローコード開発で細かい調整をする際など、専門知識が必要な場面も出てくる	留意点E
13	製品仕様が業務にマッチしなかった		アドオン開発した際の影響範囲の把握や、製品自体の仕様変更時のリスクや修正コストが上がると思われる。基本機能では、実現できない機能を追加する際の製造コスト増大。	留意点F
14	製品仕様が業務にマッチしなかった		上記のメリットの反面として、ノーコード開発製品の制約事項により表示、検索機能などに制限が発生し、運用面で補完することになった（リリース優先の処置として、発注側として合意している）。ノーコード開発製品について、性能や要件などの詳細確認を実施するとすると、かなりの時間が必要であったと想像できる。開発案件ごとに製品が異なると、管理が困難になる可能性を否定できない。	留意点A
15	開発初心者（ユーザ自身）では開発が困難だった		要求仕様に対して、カスタマイズが必要か否かの判断について、調査に時間要したり、経験に基づく製品知識が必要。	留意点F
16	開発初心者（ユーザ自身）では開発が困難だった		ノーコード・ローコード開発で何をどのような目的で開発するか次第ではあるものの、システムを作ったとしても、技術的負債をため込まないように保守性を維持しながら改良を加えて育てていくということ自体に、本質的な難しさがあるように思います。ノーコード・ローコード開発は、ソフトウェアライフサイクルの一側面のみを簡単化している文脈だと思いますので、兼局の所、ソフトウェアライフサイクルを通じてのソフトウェアエンジニアリングは、究極的には求められるとあ思います。	留意点F
17	開発初心者（ユーザ自身）では開発が困難だった		既存システムが既に構築されている状況にて、『既存機能を踏襲するようにマイグレーションする』場合、 既存システムの解析＝システム開発知識が必要になるため、ノーコード・ローコード開発に向かないのではないかと思案。 ※業務要件から完全に新たにシステム構築するケースがノーコード・ローコード開発に適している	留意点F
18	開発初心者（ユーザ自身）では開発が困難だった	26%	SalesForceの構造が色々あり、精通した開発者でしか品質を確保できない。	留意点E
19	開発初心者（ユーザ自身）では開発が困難だった		お客様側目線 ・製品を業務に合うように使うには専門知識が必要になるためユーザ内での保守・運用は困難。 ・製品利用ありきの開発になるため、製品で実現できない部分を諦める（アドオン開発で想定外に費用が掛かる） ベンダ目線 ・導入実績ありきの開発となるため、ベンダ内に有識者がいない状態となる。 ・製品導入実績有の他社に作業を任せたため、日立基準の作業プロセスに合わず手戻りが発生する。	留意点F
20	開発初心者（ユーザ自身）では開発が困難だった		開発の初心者では対応できない。また既存のシステム同等のＵＩもＣＳＳを当てるなどで実現できるものの、その場合にはメリットである期間やコストの圧縮が叶わない。導入教育が短くても対応可能な範囲が限定的で習得したスキルを転用しづらい。	留意点F
21	開発初心者（ユーザ自身）では開発が困難だった		–	留意点E
22	開発初心者（ユーザ自身）では開発が困難だった		–	留意点E
23	開発費用が計画よりも増大	6%	使用する製品によっては費用がかかることもあるのではないかと思います	留意点O
24	開発費用が計画よりも増大		–	留意点O
25	品質の問題（アドオン開発した範囲の品質）	6%	・ツールの選定、操作方法に時間を要する・複数のアプリを連携させることができることでの弊害	留意点F
26	品質の問題（アドオン開発した範囲の品質）		–	留意点F
27	品質の問題（製品自体の品質）	6%	開発有識者なら組み立てられるプログラムでも開発初心者の場合、または自分が考えているプログラムをノーコード・ローコードでは組み立てられない場合など出せる品質に限界がある	留意点E
28	品質の問題（製品自体の品質）		自由度の高い製品程、標準的な使い方から外れるとマニュアルに書かれていない制限にぶつかったり、不具合に遭遇する。	留意点F
29	その他	6%	想像の範囲で回答します。 ・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。	-
30	その他		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。	-
31	開発期間が計画よりも増大	3%	カスタマイズの要望が多かったり、製品仕様の理解が足りない人が開発を行うと、想定より大幅に時間がかかった。	留意点F

#	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

2. 直近で参画しているプロジェクトにおける役割を教えてください。（当てはまるものを全て選択して下さい）

項目	選択数	割合
①プロジェクトマネージャ	5	10%
②チームリーダ	2	4%
③開発担当	7	14%
④テスト担当	10	20%
⑤品質保証	14	28%
⑥その他	12	24%
合計	50	100%

3. 経験のある開発プロセスを教えて下さい。（当てはまるものを全て選択して下さい）

項目	選択数	割合
①ウォーターフォール型開発	28	62%
②アジャイル型開発	13	29%
③その他	4	9%
合計	45	100%

4. ノーコード・ローコード開発の経験について教えて下さい。（当てはまるものを選択して下さい）

項目	選択数	割合
①開発、品質管理の経験がある	14	45%
②経験が無い	17	55%
合計	31	100%

改訂版ノー＆ローコード虎の巻

#	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	留意点A （会社独自のローカルルールをやめるなど） ローコード・ノーコード開発に合わせた、 業務フローの見直しも必要
2	ソフトウェアの要求			留意点B 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		■他研究コースからフィードバック 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	留意点C 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	■他研究コースからフィードバック 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 ■他研究コースからフィードバック 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。	留意点D ある程度の設計知識は必要であり、開発初心者だけでは難しい
5	ソフトウェア構築	期待感C 開発初心者でもすぐに利用可能	■開発時の欠陥からフィードバック 製品仕様の理解不足により、設計不良を発生させた	留意点E ・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、 自前で内部設計・実装（構築）できるレベルの、設計力を持つメンバを配置する必要がある。 ・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保有する必要がある。
6	ソフトウェア構築	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	■他研究コースからフィードバック パッケージ製品に対して、独自の改修を行うと影響があり、一定の割合を超えると、通常のウォーターフォール開発よりも生産性・品質は悪化する。 ■他研究コースからフィードバック 製品の機能を利用するよりも、開発者が直接コードを書いた方が自由度が高く作れると判断して、改修量などはあまり考えずに追加機能を実装する事例があった	留意点F 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）
7	ソフトウェアテスティング	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	■ローコード、ノーコード開発者からのフィードバック 実際の現場バグ一覧を確認した結果、単体レベル、結合レベルの欠陥が摘出されていた。	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。
8	ソフトウェアテスティング	期待感F ローコード、ノーコード製品が自動生成した部分の品質は担保されているため、テストや保守が不要	■他研究コースからフィードバック 障害発生時にサービス提供ベンダにログ提供を依頼したものの、もらえなかったので、自分たちでログの取得を行った。 ■他研究コースからフィードバック ローコード開発で作られた、システムにおいて、割り込み制御に関する障害が発生。 根本原因の分析を実施したが、仕様がBlackBoxであたため、根本原因にはたどり着くことができなかった。	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、 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。
9	ソフトウェア保守	期待感G 保守不要	■過去担当PJの経験からフィードバック 製品のVerUpについて 追加で実装したJava Scriptで開発した部分を全て作り直す事が必要があった。データをオリジナル形式で保有しているで、VerUp作業には過去データ全てをコンバージョンする必要がある。	留意点I 互換性がなくなった場合のリスク対策が必要。
10	ソフトウェア保守			留意点J 製品のVersionUp時に発生する影響の見極めが必要。
11	ソフトウェア保守	期待感H いつでも、手軽にどの製品も使い続けることができる	■研究員の知見からフィードバック 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	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。
12	ソフトウェア構成管理	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	■研究員の知見からフィードバック 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。	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。
13	ソフトウェアエンジニアリング・マネジメント	期待感J 海外製品も自由に利用できる	■他研究コースからフィードバック ノーコード開発をしているアプリが海外製品の場合、個人情報保護の問題、欧州の規格の関係で売却する際に問題になった。	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。
14		期待感K 開発も簡単になるので、付随して個人情報の管理も楽になる。	■他研究コースからフィードバック GDPR。欧州外に持ち出すのが法に触れる。	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。（各国や各地域外に個人情報を持ち出すことが制限されているケースがあるため。）
15	ソフトウェアエンジニアリング・マネジメント	期待感L 海外製品も無料で利用可能であったり、コストを抑えることが可能	■書籍からの引用 本格運用となると有料プランを使用しなければならない傾向がある。 ・独自ドメイン ・SEO対策 ・その他、運営に関する詳細設定	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。
16	ソフトウェア品質	期待感M 顧客の要望をすぐに取り込める為、要求仕様に対する適合は高いものが作れる	■過去担当PJの経験からフィードバック ユースケースを読み込ませて作成する製品について、ユースケースには業務と適合する為の詳細な記載が無かったが、ベンダ任せとなっていた為、細かい部分の開発はどのように実施しているのか不明のまま開発を進めた。 結果、エラーが多発するなど、開発トラブルとなった。	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。