第 5 回特別講義　レポート

日時	2018 年 10 月 12 日（金）　10:00 ～ 12:00
会場	（一財）日本科学技術連盟・東高円寺ビル　地下 1 階講堂
テーマ	品質技術の実践
講師名・所属	足立　久美　氏（株式会社デンソー／本研究会 実践コース 副主査）
司会	小池　利和　氏（ヤマハ株式会社／本研究会 運営小委員会委員長）
アジェンダ	1．品質の実践とは 2．想定外との戦い 3．制約との戦い 4．プロセス改善の事例 5．課題解決の事例
アブストラクト	品質の実践を「品質技術を効果的に活かして課題解決をすること」と捉え、その品質の実践事例（考え方中心）として、次の 2 つについて紹介する。 ① 組込みソフトウェア開発におけるプロセス改善の実践事例 ② 旧第 6 分科会「派生開発」での 8 年間の活動で培ってきたノウハウをまとめた「論文構造を活用した課題解決方法」の実践事例

講義の要約

足立　久美　氏

・1982 年株式会社デンソーに入社、主にエンジン制御コンピュータソフトウェアの開発に従事。現在は、品質管理部システム品質保証室にて、ソフトウェアプロセス改善やプロセスアセスメント、機能安全活動に従事されている。Automotive SPICE　Competent Assessor。

他に、ISO／IEC／SC7／WG10 エキスパート、南山大学及び名城大学非常勤講師、日本科学技術連盟 SQiP 研究会「実践コース」副主査、同 SQiP シンポジウム委員、同品質保証部長の会企画委員としてもご活躍されている。

1．品質の実践とは？

◆実践の言葉の定義

ソフトウェア工学におけるプラクティス（Practice）とは、作業や設計に使用される効果的な技法／手法や、時間をかけて多くの人に使用されてその効果が証明されているものを言う。一方で、

よく似た言葉にパターンがあり、ソフトウェア工学におけるパターン（Pattern）とは、色々な作業や設計において、比較的共通に発生する状況、又は共通によく使用されるノウハウのことを言う。また、パターンのうちダークサイドに陥るものをアンチパターンと呼ぶ。

⇒“プラクティス”と“パターン”の関係性イメージ

［モデル（幹細胞）：方法論］　［プラクティス：方法／技法］　［パターン：ノウハウ（プチテク）］

理論的←----------→実践的

⇒“実践”の一般的な意味は、主義・理論などを実際に自分で行うこと（Goo 辞書より）とある。XDDP や USDM などの理論があれば一度実際に使ってみることと理解している。

⇒実践と理論の関係を言い表す言葉として、「実践なき理論は空虚であり、理論なき実践は無謀である」（ピーター・ドラッカー）があるが、考えるには難しいが感覚的には分かる。

◆理論と実践は、対極関係として語られるが、理論と実践の間をスムーズに繋ぐのが“練習”であると考える。

理論　⇒　練習（訓練）　⇒　実践（本番・実戦）

◆“実践”ではなく“実行”ではダメ。実行は失敗を容認する（ダークサイド）が、実践は成功を目指す（ライトサイド）活動と考える（イメージとして）。

失敗が許されない特に医学（医療現場）の世界では、医学知識⇒練習（研修）⇒実践、の流れが必要で、本番で最高の能力を発揮するために、理論と実践の間には必ず訓練（練習）が必要である。

◆Doctor X の大門未知子が失敗しない理由は何か？

⇒起こりうるすべてのケース、障害を予想して完ぺきな準備をしていたから。ただ大門未知子も想定内と想定外の領域を持っているはずで、想定外との戦いをしている。

2. 想定外との戦い

◆不具合、不祥事が起こる度に使われる言い訳として、“記憶にございません”、“想定外でした”が常套句だが、物事には、「想定外」で済まされないものがある。容易に想定できるものも「想定外」といって逃げる人が多い。このような人たちは反省しないので、同じことが繰り返される（再発）。

◆想定外の例

⇒四国高松市　琴電高松築港駅　駅真横の高松城のお堀にいる魚は鯉ではなく鯛。お堀が海。

⇒アメリカ　想定外の 2.7m、470kg の巨大イノシシが捕獲された。

⇒中国　常識を外れた過積載のトラック。

⇒インド　常識を外れたバケツを過積載しているオートバイ。

⇒日本　東電　福島第一原発事故　電源喪失に備えた電源設備（2 系統）が地下に設置されており、津波で水没。

◆“想定の範囲（想定）”と“想定の範囲外（想定外）”の中間にある“想定すべき範囲（予見可能範囲）”は想定外といえるのか。製品に問題があったときの裁判所の考え方で言えば、本来の使い方（想定）に対して、誤った使い方が予見可能範囲であれば想定外とは言えず、そこで発生した問題は想定不十分となる。

想定外　＞　予見可能範囲（予見範囲含む）⇒　そこで発生した問題　＝　想定不十分

3．制約との戦い

◆実践とは制約（矛盾）との戦いである。6 つのポイントで説明する。

①実践は理論通りにはいかないことが多い

⇒“課題”に対する“結果（答え）”を導き出す際、何等かの“理論（方法論）”を使って“実践”することになるが、この実践には必ず“制約”条件を伴っている。実践とは制約の中で成り立つ答えを導き出すことである。

⇒例）理論：NOT 回路（反転）

実践：理論通りギリギリの設計では市場で不具合が出る。

リアルな世界では遅れ／バラつき／イズがあり、非機能要求の制約を盛り込んだモノづくりが必要。

②矛盾を解決することがエンジニアリング

⇒作用すれば反作用がある。薬には副作用がある。陽が当たれば、陰になる所がある。

⇒例 1）愛知県田原町　表浜（小島海岸）　消波ブロック

砂浜の浸食→消波ブロック設置→海ガメ産卵の妨げ→海ガメの減少（絶滅の加速）

砂浜の浸食→放置（不作為）→砂浜の浸食の拡大→海ガメの産卵場所の減少→海ガメの減少

⇒消波ブロックを設置しても設置せず放置しても、いずれも海ガメ減少に繋がるジレンマ。

⇒消波ブロックを全面設置から部分設置に変え、砂浜浸食の抑止と海ガメ産卵場所確保の双方を両立。

⇒Sweet Spot Thinking（SST）

メリットとデメリットの折りあいをつける。

許容範囲　＞　最小工数　へ近づけて行くシステム思考をする。

⇒例 2）愛知県田原町　表浜（小島海岸）堆砂垣（たいさがき）

砂浜の浸食→堆砂垣（たいさがき）部分設置⇒砂浜浸食の抑止と景観確保の双方を両立。

⇒例 3）ネックストラップの事故防止装置

強い力が加わると事故防止パーツが分離してストラップが外れる仕組みであるが、強度が強すぎると人命に関わり、弱すぎるとセキュリティに関わるという、相矛盾する問題を解決。

③要求はある意味制約である（特に非機能要求）

⇒要求には機能要求と非機能要求がある。

機能要求：製品やサービスが遂行すべき働き（機能）に対する要求

非機能要求：製品やサービスが遂行すべき働き（機能）の良さや制約に対する要求

例）機能の発揮方法、性能、信頼性、使い勝手など

⇒非機能要求が決まらないとアーキテクチャ（手段の基本構成）が決まらない。

例）要求：地点 A から地点 B に行きたい

機能要件：人と荷物を乗せて移動する

非機能要求：どこまで？（移動距離）、いつまで？（移動時間）、いくらで？（移動コスト）、安心できる？（安全性）

アーキテクチャ：移動手段　空を飛ぶ（飛行機）、海を渡る（船）、陸路を走る（車）

④実践には複眼を持つことが大切　-複眼思考の勧め　複眼は福眼に通ずる-

⇒あるワークショップから「テスト」における課題とその解決方法について議論して分かったこと。

（1）テスト工程の中だけで全ての課題の解決が困難である。（プロマネ、上流工程の課題が多い）

（2）開発プロセス全体を対象とする複合視点、システム思考での課題解決が必要である。

（3）「総合診療医」のような製品開発全体を診療する「総合エンジニア」が必要である。

⇒以下のような背景がある。

ソフトウェア開発の大規模化・複雑化→分業の促進（自分の担当しか分からない）→狭い範囲での解決困難化

※ありきたりの常識や紋切り型の考えに囚われずに、広い視野から多視点で物事を捉える必要がある。

⇒総合診療医（ドクターG）とは、病気を心身から全体的に診療する医師。病気の予防にも携わる。患者の特定臓器に着目するのではなく、全体的な健康問題に向き合って治療を行う。

診療　＝　診察　＋　治療

診察：医師が患者の体を調べて病状・病因などを探ること←課題形成

治療：病気やケガを治すこと←課題解決

⇒総合エンジニアの提言：総合エンジニアとは、製品開発における特定のプロセス（技術）に着目するのではなく、開発プロセス全体を複眼で捉え、システム思考（※）で課題解決を行うエンジニア（エンジニア G と呼ぶ）。

※システム思考と、物事のつながりを構造として捉えること。

CO2 増加⇒地球温度上昇⇒北極の氷現象　－因果律（時系列的連鎖）－

⇒どんな装置でも得手／不得手があり、いくつかの装置を組み合わせて用いることで補完している。

例 1）自動運転車

カメラ、ミリ波、ライダー、クリアランスソナーといった複数のセンサーを組み合わせて実現する。

（同じセンサーでも搭載場所によって使い方や用途が違ってくる）

例 2）ガン検診

レントゲン、CT スキャナー、PET を組み合わせて診断する。

⑤実践力　－ある制約条件のもとで、価値を生み出す力-

⇒実践の 6W。6 ワークで、価値（Value）を生み出す。

(1)ヘッドワーク：よく考えて行動する。（但し、タイミングを外すな）

(2)チームワーク：チームを組む。（一人でやれることには限界がある）

(3)フットワーク：やると決めたら即行動する。

(4)ネットワーク：外部の情報網、支援網を活用する。

(5)フィールドワーク：問題は現場で発生している。（`現地`に行き・`現物`を見て・`現実`を知る）

(6)不惑：迷わない。（一度決めたことはやり遂げる）

⇒不惑については、「やりたいこと」、「やるべきこと」、「やれること」の関係性を識別し、どう折り合いをつけるかが大事となる。目的「やるべきこと」と現状「やれること」の差の GAP、「やるべきこと」と「やりたいこと」のベクトルの GAP の認識が必要。

⑥アンチパターンは実践のポイント

⇒我々の周りに潜在しているアンチパターンを切り出して、戒め・警句、改善のポイントとする。

⇒例）「憲章炎」患者の実例

ある会社の機能安全担当者からの質問で「アジャイルプロセスの採用で困った」こと。

アジャイルプロセスの採用が決定→アジャイルではドキュメントは書かない（※1）→機能安全ではアカウンタビリティが重要→エビデンスが必要（※2）

⇒（※1）と（※2）で矛盾が発生。アジャイル憲章は、「ドキュメントを書かなくてもいい」とは言っていない。必要なドキュメントは後になっても必ず作るのが正しい。

名称：憲章炎（けんしょうえん）

内容：アジャイル憲章を正しく理解していないこと。

対策：ダークサイドに落ちた人にアジャイル憲章を正しく教える。

4. プロセス改善の事例

◆欧州でのプロセス実装事例

⇒LLD（Low Level Design）にアジャイル開発を適用している。

HLD（High Level Design）は古典的な「反復型開発モデル」のまま。開発者をアーキテクト（HLD 対応）、設計者（LLD 対応）、テスターに区分している。

アーキテクトが設計者の仕事の管理をしている。

⇒ツールによる設計支援が充実している。

ツールの中で全ての仕事をしている。ツールが統合されて、すべてのタスクをカバーしている。

ドキュメントを作成する必要がない。必要なドキュメントはツールが吐き出してくれる。

但し、過度なツール依存は別のリスクを持っており、ツールが止まれば全ての SW 開発がストップすること、

ツールのエキスパートチームが日々ツールの保守・改良をする必要がある。

◆プロセス改善あるある“8”

⇒製品開発の 5 大リスク（PASSO）。

Process：製品開発プロセス能力の要求に対応しなければならないリスク。

AT：Advanced Technology：人口知能（AI）、機械学習などの先端技術の導入によるリスク

Safety：機能安全の国際規格 ISO26262 対応を充分な体制を構築し満足しなければならないリスク

Security：セキュリティがセーフティに大きく影響を与えるリスク

OSS（COTS）：OSS のライセンス違反や OSS の選定を誤ったことに対する訴訟リスクや製品の改修リスク

⇒Process は他の 4 つの基盤技術であり、その整備が肝である。不完全な開発プロセスのもとは、設計活動はうまくいかない。

⇒プロセス改善の 8 つのアンチパターン（A-SPICE をベース）

プロセス改善がうまくいかない理由には多くの原因が考えられるが、特に目立つ共通の 8 つのアンチパターンを紹介する。

（P.1）視野狭窄

症例：A-SPICE の適用範囲の誤解。

症状：プロセスの構築と改善活動が不十分なので OEM によるプロセス監査の準備は間に合わない。

システム／ハード／ソフト間の連携した活動が不完全である。

原因：A-SPICE はソフトウェアのみに適用されると誤解している人が多い。特にシステム技術者。

システム／ハード／ソフトの各ドメインが個別最適活動をしているため、製品（システム）として品質保証するという意識が希薄である。

処方箋：A-SPICE の”plug-In” concept を正しく理解する。

（P.2）レベル崇拝

症例：プロセス改善活動の目的がプロセス能力レベルの認定

症状：認定された A-SPICE の能力レベルより実際のプロセス能力のほうが低い。

組織の標準プロセスが開発チームの現状と合わない。

原因：目標のプロセス能力レベルを取ったら、プロセス改善活動を止める。

プロセスは定義された時から陳腐化する。（いつかテーラリングでは吸収できなくなる）

処方箋：開発環境の変化に対応するために、継続的なプロセス改善活動を可能とする仕組みを作り運用。

レベルの取得は手段であって目的ではない。プロセス改善の目的は、設計の品質と生産性の向上である。

（P.3）支援過多

症例：一方的支援と一方的依存

症状：改善スタッフは開発チームにとって過剰又は不十分などの不完全なプロセスを開発チームに提供。

改善スタッフから提供されたプロセスを遵守しても、設計の生産性と品質向上が難しい。

原因：改善スタッフは、開発チームの実情を考えることなく、開発チームに対してプロセス改善活動を一方的に支援している。（一方的支援）

開発チームは「改善スタッフがプロセス改善活動の担当である」と考えているので、開発チームは自主的にプロセス改善活動をすることがない。（一方的依存）

処方箋：開発チームはプロセス改善活動の主体であり EPG は開発チームのプロセス改善活動を支援するものであると認識すること。支援のための重要な観点は自発性を育てること（自分で考えて行動することが大切）。

（P.4）“why”の蒸発

症例：ルールの意図やチェックの目的の欠落

症状：ルールやチェックリストの中に、なぜこのルールが存在しているのか、またなぜこのチェックをしなければならないのか、その理由が理解できないものがある。これが、不十分なチェックやルール遵守を誘発し、品質低下の一因となっている。

原因：組織標準やチェックリストには、ルールやチェック項目（What, How）が記述されている。しかし、ルールやチェック項目の存在理由（why）が記述されないことが多い。また文書を簡略化（洗練）すると、後で「なぜそうなったのか」が分からなくなる情報欠落のリスクがある。

処方箋：管理のための組織標準やチェックリストに、ルールやチェック項目の存在理由（why）も記述すること。後で保守メンテできるようにする。

（P.5）PPT／EXCEL 症候群

症例：開発ツールが EXCEL に偏重し過ぎである

症状：構成管理、変更管理、問題管理、トレーサビリティ管理に多くの工数が必要であり非効率である。また、ミスが発生しやすいので信頼性が低い。

原因：EXCEL が利便性の高い専用ツールの代わりに、管理ツールとして多く使用されている。したがって、管理が属人的になっている。

処方箋：管理のための専用ツールを使用して属人性を廃する。WORD（正式文書）／EXCEL（表計算）／PPT（プレゼン）の本来の使い方を理解して利用する。属人的であることがリスクであることを認識し、ツールを使っていないこと自体、非効率かつ作業品質が悪い証拠である。改善がうまくいかない理由として、十分な装備・ツールを与えずに PPT／EXCEL だけで無理なルートを綱渡りしている。

（P.6）ノウハウの蒸発

症例：アウトソーシングへの過度な依存

症状：設計者の設計スキルやノウハウが低下するので、設計変更や問題が発生した場合、設計者はそれに対応できなくなってしまう。

原因：設計をする機会が減少するので、スキルやノウハウが蓄積されない。特にアウトソーシングの多重化と“丸投げ”が良くない。

処方箋：1）組織の取得戦略（内製／外注の判断基準）を定義する。

2）外注先を管理し、“丸投げ”をしない。丸（○）投げは、“バツ（×）投げ”。
アウトソーシングはコントロールされ、かつ管理されなければならない。

3）アウトソーシングの多重レベルを制限する。
多階層になるほどノウハウが蒸発するので品質コントロールが難しくなる。

（P.7）テーラリング不全

症例：テーラリングが不完全（うまくできていない）

症状：人によってテーラリング方法がばらつき、テーラリングされたプロセスのタスクに過不足が発生し、プロセス品質が安定しない。

原因：テーラリングガイドラインが不十分（又はそれが存在しない）、又はテーラリングの理解不足。

処方箋：1）テーラリングに対する正しい理解を持つこと。

テーラリングとは、プロジェクトによって仕事の内容、制約条件が違うので、製品特性及びプロジェクト特性をパラメータとして、テーラリングガイドラインに従って、組織の標準プロセスをプロジェクトに実際に適用するプロセスに変換することである。

2）十分なテーラリングガイドラインを定義し、運用すること。

プロジェクト毎の勝手なテーラリングを許すと、プロセスの管理を難しくする。

テーラリングガイドラインを用いて、テーラリングの管理に制約を設ける。

（P.8）部分最適

症例：機能間協調の乱れ

症状：手戻りや不具合の発生による、製品開発の開発効率や品質の低下。

原因：システム／ソフトウェア／ハードウェアの設計間の連携した活動が不十分（マネジメント層がマネジメントしていない）で、各々が自己中心的な活動をしている（部分最適）。

処方箋：1）システム／ソフトウェア／ハードウェアの各設計部署間の役割分担の明確化、及び必要十分な組織間インターフェースの確立。（例えば合同の進捗会議や合同の検証活動の充実）

バラバラな活動は開発効率の悪化と品質の低下を招く。

2）システム設計／ソフトウェア設計／ハードウェア設計間の統括管理の　実施（全体最適）

⇒あなたはどの世界に住んでいますか？

[天上界：ボヤすら起きない] [人間界：ボヤで済む] [修羅界：火消しで何とか鎮火] [地獄界：そこら中 火の海]

品質リスク Low←----------→品質リスク High

プロセス改善は状況（住んでいる世界）によっては有効とは限りません。状況によっては外科手術も必要です。

1)プロセス改善は体力の増進に相当します。

状況を良く見定めて、適切な対処をすることが大切です。

2)プロセスは定義された時から陳腐化が始まる。

継続的なプロセス改善をして、プロセスの老化を防止しよう！

この時にプロセス改善のアンチパターンが役に立つでしょう。

3)アンチパターンは他にもたくさんあります。

アンチパターンに気を付けて、継続的改善活動を推進してください。

5. 課題解決の事例

◆SQiP 研究会「派生開発」分科会（旧第 6 分科会）

25SQiP（2009 年）~32SQiP（2017 年）の 8 年間、派生開発における課題形成力と解決力の育成の場として活動。

⇒派生開発の様々な問題を取り上げ、その解決策を考案する。

派生開発における問題として、変更箇所、変更の影響の特定に起因する問題が多い。

活動の進め方として、以下の手順で課題形成と課題解決を行う。

メンバーの抱える問題や経験の整理→共通課題の明確化→解決方法の考察→解決方法の効果の検証

⇒問題解決のために、XDDP・USDM・PFD・論文思考 などの手法を効果的に活用する。

「XDDP」機能追加と変更を異なるプロセスで対応する（追加機能要求仕様書、変更要求仕様）

「USDM」要求に理由を付けて仕様を階層構造で表現する

「PFD」多様な要求を満たすための合理的なプロセスを設計する

「論文思考」論文とは課題形成／解決の一連の過程とその成果を論理的に文章として纏めたもので、これをフレームワークとして課題形成と課題解決を行う

⇒8 年間の活動成果として、

研究員数：75 名、報告書：18 件、最優秀賞：6 件、優秀賞：3 件、SQiP シンポジウム採録 7 本。

◆論文構造を活用した課題解決方法

論文を書こう。論文構造を活用した課題解決方法を提案する。

⇒論文とは？

論文と聞くと壁が高くて登れないイメージがあるが、階段があれば登れるのではないかと考えた。

論文の定義「課題形成と課題解決の一連の過程とその成果を論理的に文章としてまとめたもの」

→論文の章・節構造が階段の役目を果たせるのではないか。

→論文を書きながら課題形成・課題解決することができる。

⇒課題形成／解決モデル（WH-2Why）

問題と課題の違いは何か？

問題：現状とあるべき姿との間にあるギャップ。現在発生している好ましくない状態、不都合のこと。

課題：問題（ギャップ）を解決するために取り組むもの。目的を達成するために、解決すべき問題。

現状 → 現状の把握 → 問題（抱えているもの） → 課題の選定 → 課題（解決すべき問題）

課題形成とは：現状抱えている問題の中から、課題を決定すること。

課題解決とは：課題に取り組み、ゴールを達成すること。

課題形成／課題モデル（WH-2Why）とは、課題形成／解決を What, How, Why でモデル化し、起承転結をマッピングしたもの。

課題形成：「起」現状抱えている問題の中から

「承」解決すべき問題（What）←何故その課題なのか？（Why）

課題解決：「転」具体的な解決方法（How）←何故その方法なのか？（Why）

「結」成果

課題解決シナリオは、課題の形成から解決までの一連の流れ（筋書）を「起承転結」に分けて作成したもの。

その筋書は、概要レベルから段階的詳細化しながら練り上げると良い。また起承転結のバランスと一貫性が大切である。

⇒「論文フレームワーク」とは

論文の典型的な章・節構造を定義し、WH-2Why モデルをマッチングしたもの。

＜構成＞

タイトル、著者・所属、概要（要旨）、本文（※）、謝辞、参考文献

（※） 本文の構成

1 はじめに　論文全体の説明（前書き）

2 解決すべき問題　何を解決したいのか?（目的）　What?

＜課題形成＞　何故解決したいのか?（理由）　Why?

－現状分析　抱えている問題の把握　＜起＞

－課題提起　課題（解決すべき問題）の決定　＜承＞

－先行研究　先行研究の調査（結果）

3 解決策の提案　どのように解決するのか？　How?　＜転＞

＜課題解決（仮説） なぜその手段を選んだのか？（考え方と具体的な方法）Why?

4 解決策の評価

＜課題解決（実証）＞

－評価方法　　解決策の評価方法

－評価結果　　解決策の評価結果

－結果の考察　評価結果から得られた結論（知見・ノウハウ）←まとめには入れない

5 まとめ　研究会全体の総括（全体のまとめ）　　＜結＞

－成果

◆今後の進め方

＜論文フレームワークのアーキテクチャ＞

「タイトル」→「概要」→「本文」の順に、段階的詳細化する。

「タイトル」と「概要」と「本文」の間には一貫性があること。

「本文」の 2 章は 課題形成、3 章から 5 章は 課題解決 のプロセスとして表現できる。

論文フレームワークを課題形成・課題解決のテンプレートとして活用できる。

＜論文フレームワークの使い方＞

論文フレームワークは V 字を形成している。

2．解決すべき問題　←――→　5．まとめ

＼　　／

3．解決策の提案　←→　4．解決策の評価

＼／

小分けして評価しながら仕上げていくこともある。

解決したいことから始まることもある。

手戻りを繰り返しながら進むことが多い。

論文フレームワークは課題形成／解決手法である。

⇒良い論文を書くためには

論文にも品質があり、ネタが新鮮（新規性）・美味（有用性）・安全（信頼性／論理性）であっても、それを味わって（理解して）もらえなければ意味がない。→　論文フレームワークは理解容易性の向上を助ける。

＜論文の作成に必要な 3 つの力＞

1)論理力：論理的に考える力

因果律（物事は原因と結果の連鎖）を意識して書きましょう。

2)文章力：分かりやすく文章表現する力

読んでもらえる論文を書きましょう。

3)客観力：客観視する力

手前味噌にならないように注意しましょう。

＜力を得る方法＞

・論文のフレームワーク（定石）に従って書く：守破離。最初は型を覚えてそれを守ることが大切

・良い先生に指導してもらう：優れた指導者のもとで訓練に励むことが大切

・先人の良い論文を読んで学ぶ：論文を選別（タイトル、概要、前書き）して読む。

・論文の査読結果、発表時の質問から学ぶ

＜モデル化（抽象化）の重要性＞

・ベストプラクティスの他への適用の容易化を可能とする。一旦モデル化（抽象化）してから適用（個別化）する。

他でうまくいったやり方をそのまま持ってきても、うまくいかないことが多い。

・構築したモデル（自分のアイデア）にネーミングして Identity を持たせる。

ネーミングは提案を活かしも殺しもする。

特徴や性質が思い浮かび、一般常識や習慣から外れない、短くて簡単な覚えやすい名前を付ける。

＜論文作成の躓きどころ＞

・新規性の確認でとまどう（自分が最初？）

自分が独自で考えたアイデアであっても、すでに他の人が同じアイデアを出している場合がある。

先行研究調査を実施する。アイデアをメモするときは誰のアイデアか、出典をメモし引用可能とする。

・解決策の評価でとまどう

評価方法が分からない時は、先行研究・関連研究から良いヒントを得る。

改善効果が分からない時は、改善前の状態（before）を予め把握しておく。

・考察とまとめを混同する

考察とまとめの違いを確実に理解する。

＜RQ と GQM について＞

・RQ（Research Question）とは？

研究テーマ（課題）に答えるために設定される質問の集まりで、この質問（RQ）に全て答えれば研究テーマは達成（課題は解決）される。

課題を複数の小課題に分類すると解決し易くなる。大きな課題になればなるほど課題の分割（段階的詳細化）は有効になる。

・GQM と RQ を用いた研究アプローチとの類似性

GQM では、計測はやみ雲に実施せず、まずゴールを決めること。

ゴール達成の基準（質問）を決め、情報を集めこれに答える。

RQ は GQM と対比させると分かりやすい。

⇒まとめ（論文思考）

「論文思考」とは、WH-2Why モデルに基づいた論文フレームワークを使って、課題形成／解決をすること。

「論文思考」を使って、課題解決をしよう。論文を書いて、発表し、人の意見を聞こう。

論文を書いて発表するメリットは、

論理的思考ができる、発表がうまくなる、実績が残る、見識が広がる、仲間が増える、自分の Positioning ができる、講演依頼が来る。

自分の考えをもっていると、それが引力を持ち、それを披露するチャンスが向こうからやってくる！

論文は新たな飛躍のパスポートである。