カリキュラム再設計 構想書

清風情報工科学院 システム専攻 カリキュラム再設計 構想書

本文書は、2025年2月のChatGPTおよびClaude（クロ）との対話を通じて構築された設計思想と議論の記録である。 システム専攻をモデルケースとして設計し、他専攻（ゲーム・広告デザイン）への展開の起点とする。

【共通基盤】

第1部：設計思想

1-1. なぜカリキュラムを再設計するのか

従来のITカリキュラムは「コーディングを教える」ことに最適化されてきた。学生にプログラミング言語の文法を教え、動くものを作らせ、就職に必要なスキルを身につけさせる。この枠組み自体は間違っていなかった。

しかし、生成AIの登場によって前提が変わった。

- AIがコードを書く時代に、「コードが書ける」だけでは差別化にならない
- AIが出したコードを「動いた、OK」で終わる学生が増えている
- 特定言語の深い知識は、AIに聞けば出てくる時代にはアドバンテージにならない
- 一方で、概念レベルの理解（認証とは何か、トランザクションとは何か）は、AIにショートカットさせられない

就職市場も変化している。少子化で売り手市場が続く見込みだが、AIの浸透で「新人に求めるハードル」が上がる可能性がある。留学生市場では既に日本語能力への要求が上がっている。IT分野の日本人採用でも同じことが起き得る。

企業が本当に求めているメッセージは「基礎ができている人を送ってください。言語は社内で教えます」である。表面的な即戦力（Java書けます）を装うより、本当の即戦力（何を渡されても立ち上がれる基礎力）を育てる方が、学生にとっても学校にとっても正解である。

本再設計の目的は、「特定の言語やフレームワークに依存しない、概念レベルの理解を持った人材」を育てるカリキュラムを構築することにある。

1-2. 「概念の肉体化」とは何か

本構想書の中心にある考え方は「概念の肉体化」である。

概念の肉体化とは、**用語を知っている状態ではなく、自分のコーディング体験や実務体験と結びついて血肉になっている状態**を指す。

例えば「認証」という概念：

- **知っているだけの状態**：「認証はユーザーが本人であることを確認すること」と言える。しかし、実装したことがないので、セッション認証とトークン認証の違いを体で分かっていない
- **肉体化された状態**：PHPでセッション認証を作った経験があり、JavaScriptでトークン認証も作った。だから「この2つは目的は同じだが仕組みが違う」と実感で分かる。新しいフレームワークで認証を実装するときも「はいはい、これね」となれる

肉体化は、以下のサイクルで起きる：

1. **やってみる**（具体的な体験）
2. **言葉にする**（自然言語で説明する）
3. **別の文脈でもう一度やる**（異なる言語・異なる場面）
4. **比較する**（「前回と今回は何が同じで何が違うか」）

このサイクルが回って初めて、概念は肉体化される。1回やっただけでは身につかないし、やるだけで言葉にしなくても身につかない。

1-3. AI時代に陳腐化しない芯とは

AI時代に価値が下がるもの：

- 特定言語の文法知識（AIに聞けば出てくる）
- 定型的なコーディング作業（AIが代替する）
- 特定フレームワークの使い方（ドキュメントをAIが読む）

AI時代に価値が上がるもの：

- **概念レベルの理解**：認証、トランザクション、状態管理、API設計などの「なぜこうなっているか」の理解
- **データ構造の設計力**：何をどういう形で持つか、保存するか、動かすかを決められる力
- **自然言語の分解力**：業務の言葉をデータに落とせる力（後述）
- **AIが書いたコードを読み解く力**：「どこが本質でどこがおまじないか」を判断できる力
- **セキュリティの視点**：「攻撃者ならこれをどう悪用するか」を考えられる力
- **設計書類を残す力**：「何を作りたいか」を言語化してAIに渡せる力

本カリキュラムは、これらの「陳腐化しない芯」を3年間で育てることを主眼とする。

1-4. 帰納的学習の原則

本カリキュラムは、**帰納的学習（体験から概念を引き出す）**を基本原則とする。

演繹的学習（先に理論を教えて適用させる）は、清風の学生層には効きにくい。抽象概念を先に渡されても、それが何を意味するか体で分かっていないから使えない。

帰納的学習では：

1. まず具体的な「困った体験」をさせる
2. 何度か異なる課題で同じ壁にぶつからせる
3. 学生自身が「こうすればよかったのか」と気づく
4. その気づきに**後から名前（概念ラベル）を与える**
5. 別の文脈で応用させ、概念を強化する

これはまさに慣習法の形成過程と同じ構造である。先に法律（理論）があるのではなく、具体的な事例の蓄積から原理が浮かび上がってくる。

この帰納的学習はAIとの協働と非常に相性が良い。学生が困ったときにAIに聞く。AIが答えを出す。しかし「なぜその構造なのか」は学生自身が体験から理解していないと、次に別の場面で応用できない。AIは答えをくれるが、帰納的な気づきは自分の中からしか生まれない。

第2部：教養OSとの統合

2-1. 教養OS 4つの型との接続

「教養OS」は、本学院で設計中の思考力・言語化力の基盤カリキュラムである。以下の4つの型と基盤レイヤで構成される。

【基盤】語彙の5段階 ＋ 読解・要約の基礎力
　↑
【描写の型】見たものを正確に言葉にする
　↑（すべての型の入力精度を決める）
　├――【構造の型】定義・因果・比較・代償
　├――【物語の型】盛り上がる話（緩急・期待・裏切り・回収）
　└――【修辞の型】ごっこ視点で相手に合わせて表現を変える

年間構成：

- 前期前半（7回）：構造の型
- 前期後半（7回）：描写の型
- 後期前半（7回）：物語の型
- 後期後半（7回）：修辞の型
- 通年：問答ゲーム（毎日・各授業冒頭でAIテキスト問答）

これらはプログラミング教育と直結する：

教養OSの型	プログラミング教育での応用
構造の型（定義・因果・比較・代償）	要件定義、データ設計、技術選択の根拠説明
描写の型（観察→言語化）	バグ報告、現状の正確な記述、要件ヒアリング
物語の型（盛り上がる話）	障害報告、ユーザーシナリオ、処理フロー説明
修辞の型（ごっこ視点）	仕様書を誰向けに書くか、エラーメッセージの設計

2-2. 自然言語の分解力 → データ構造設計

本カリキュラムが最も重視する概念は**データ構造の設計**である。

データ構造の設計には3つのレベルがある：

- **レベル1：データをどう持つか**（変数・配列・連想配列）
- **レベル2：データをどう整理して保存するか**（テーブル設計・正規化）
- **レベル3：データをどう動かすか**（API設計・JSONの構造・どこに何を持つか）

しかし、データ構造の設計には前提条件がある。**自然言語の分解力**である。

正規化理論を教えても、日本語側に言葉の定義や粒度がないとデータを分解できない。例えば「住所」を1つのフィールドにしてしまう学生は、正規化が分かっていないのではなく、「住所という言葉が実は都道府県・市区町村・番地・建物名という複数の情報の塊である」ことに気づけていない。

これは教養OSの**構造の型（それは何か）と描写の型（正確に言葉にする）**の問題である。

分解能を増やすことと言葉を増やすことはほぼ同義であり、語彙が増えれば分解の精度が上がり、分解ができれば設計ができる。

2-3. AI壁打ちによる個別指導の代替

自然言語の分解訓練は、従来は個別指導でしかできなかった。1人の教員が1人の学生の横について「その『顧客情報』って具体的に何と何？」と対話する必要があった。コストが合わず、諦めてきた部分である。

AIの活用でこれを突破する。

LLMは自然言語の分解訓練に最適な壁打ち相手である：

- 学生が「顧客情報」と書く → AIが「具体的に何が含まれますか？」と聞き返す
- 学生が「名前と住所と電話番号」と答える → AIが「名前は姓と名を分けますか？」とさらに掘る
- 30人同時に、各学生のペースで、何度でも付き合える

プログラミング授業の中に、コードを書く前の「日本語タイム」を15〜20分入れる。AIと対話しながら業務の言葉を分解し、最終的にテーブル設計にまで落とす。この結果がそのままコーディングの題材になる。

教員はサイクル全体を設計・監督する役割に専念し、個別の壁打ちはAIが担う。

第3部：語彙領域の設計

3-1. 第1領域：金融・商取引（全専攻必修）

分解能を増やすためにどんな分野の言葉を増やすのが学生に最も有利かを検討した結果、**金融・商取引**を全専攻共通の第1領域として選定する。

選定理由：

- **転用範囲が広い**：どんな会社でも売上・原価・利益・請求・支払いがある。金融・商取引の語彙が分かる人は、どの業界のシステム開発でも業務の話についていける
- **プログラミングの題材として自然**：CRUDの王道題材（注文管理、在庫管理、顧客管理）が全てこの語彙圏にある
- **データ構造設計の訓練に最適**：注文と商品と顧客の関係は、正規化を教えるのに最も自然な題材
- **ビッグデータ処理にも接続**：売上データ、在庫データ、顧客データの分析は、この語彙が基盤になる

導入は「お店の経営」から始める。ラーメン屋の注文管理から始めて、仕入れ、原価計算、売上管理、在庫管理と広げていく。学生にとって身近な「お店」の話をしているうちに、金融・商取引の語彙が自然に身につく。

3-2. 各専攻における第1領域の意味

金融・商取引の語彙は、専攻によって異なる実践的意味を持つ：

専攻	金融・商取引語彙の意味
システム	業務理解の基礎、業務システムの基礎、ECサイトの基礎、自分自身のビジネス設計
ゲーム	ゲームの収益化（課金設計、LTV、ARPU）、他校との差別化、自分自身のビジネス設計
デザイン	業務理解の基礎、広告の目標（ROI、コンバージョン）、ECサイトの基礎、自分自身のビジネス設計

特にゲーム専攻にとって、ビジネス語彙は「保険」ではなく「本命の武器」である。ソーシャルゲーム運営で飛び交うDAU、ARPU、課金率、リテンション、LTVは全て金融・商取引の概念の応用である。さらに、ゲーム会社に入れる学生は一握りで、多くはIT企業やWeb系に就職する。そのときビジネス語彙がなければ業務系の仕事に対応できない。

3-3. 専攻別第2領域

専攻	第2領域	接続先
システム	製造業DX	3年組み込み、インド連携、IoT
ゲーム	エンタメ	物語の型、UX設計、シナリオ
デザイン	健康・ウェルネス	広告題材、「健」との対応

3-4. ビジネスOS検定との連動

本学院で設計中の「ビジネスOS検定」は、以下の5ユニットで構成される：

1. **商売と信用のきほん**：取引の基本、Win-Win、説明責任、信用

2. **お金の流れと値段のつけ方**：売上・費用・利益、原価と売値、採算
3. **モノと時間の流れ**：在庫、資金繰り、回転
4. **信用を増やす話し方・ふるまい方**：ビジネスコミュニケーションOS
5. **商売を育てる**：利益の使い道、再投資、役割分担

これらの語彙は、プログラミング教育のCRUD題材と直接対応する。ビジネスOS検定で学んだ「注文」「商品」「在庫」「顧客」の概念が、そのままデータベースのテーブル設計の題材になる。

石田梅岩の石門心学や船場のことわざによる商道徳教育とも接続し、単なるスキル教育を超えた人格教育に繋がる。

3-5. 「徳・健・財」との対応

本学院の教育方針「徳・健・財」と語彙領域の対応：

教育方針	語彙領域	具体的接続
財	金融・商取引（第1領域）	ビジネスOS検定、商売感覚
健	健康・ウェルネス（デザイン第2領域）	広告題材、IoTヘルスケア
徳	教養OS全体 + 商道徳	構造の型、修辞の型、石門心学

【システム専攻 カリキュラム設計】

第4部：プログラミング言語の選択と配置

4-1. 1年：PHP（確定）

1年前期：PHPをただの言語として使う + Linux基礎

- Webインターフェースは一切考慮しない
- コマンドラインで動く、昔のC言語のような扱い
- ここでCRUDをしっかり叩き込む
- 同時並行でHTML/CSSを別授業で進める

- Linux基礎（cd, ls, cat, grep, パイプ, リダイレクト）をPHP授業の中で並走させる

この設計の狙いは**認知負荷の分離**である。言語の文法とWebの仕組みを同時に教えると、何がどこで動いているか（クライアントかサーバーか）で学生が混乱する。分けて教え、それぞれが定着してから統合する。

Linux基礎を1年前期に入れる根拠：

PHPをコマンドラインで動かしている時期に、学生は既にターミナルを使っている。そこで並行してLinuxの基礎コマンドとパイプを教える。パイプの概念（`cat file | grep "ERROR" | sort | uniq -c`）は「データの流れ」を目に見える形にする。この概念はGitHub Actionsの中身であり、Dockerの基礎であり、AWS上のサーバー操作の前提である。2年のAWS授業でSSHでEC2に入った瞬間、3年でRaspberry Piに入った瞬間、「あ、1年でやったやつだ」となる。

1年後期：サーバーサイド言語としてのPHP

- 前期でPHPの文法は定着済み
- 前期でHTML/CSSも終わっている
- だから「クライアントとサーバーの区別」に集中できる
- PHPでのサーバーサイドWeb開発（セッション、DB接続、MVC概念）

PHPを1年に配置する教育的根拠：

- 文法が平易で、成果物が早く出る
- Javaを最初にやると、クラス・型・コンパイル・IDE設定で折れる学生が出る
- 「動くものを作る成功体験」を早く与えることは、専門学校では極めて重要

4-2. 2年：検討中（複数案）

2年目の言語選択は未決定である。以下の候補がある。

案A：JavaScript（フロントエンド中心）

- ブラウザ上のJS、DOM操作、fetch APIでPHPサーバーと通信
- 1年でPHPでサーバーサイドを理解した学生が、クライアント側を体験する
- クライアント/サーバーの概念が肉体化される
- LLMとの相性も良好（Pythonほどではないが十分）
- AWS授業（PHPアプリのクラウドデプロイ）と並走した場合、「PHPサーバーにJSのフロントからAPIで叩きに行く」という全体像が描ける

案B：Java（Spring Boot）

- 就職担当の「Java案件が多い」に直接応える

- 静的型付けの概念が入る
- 3年に組み込み（C言語）を入れる場合、Javaの厳密さが橋渡しになる
- ただしAI協働学習との相性はやや悪い（ボイラープレートが多く、AIが出すコードの「どこが本質か」が見えにくい）
- 「AIにJavaを書かせて、読んで理解して修正する」という教え方で対応可能

案C：Java（前期）+ 自然言語設計（後期）

2年目の目的を「言語習得」から「AI協働設計力の獲得」に再定義する案。

前提にある問い：学生の卒業時には、現在のプログラミング言語はかつてのアセンブラのような位置づけになるのではないか。LLMがコンパイラとなり、コードは人間が理解できなくても（理解できれば解像度は上がるが）、自然言語でプログラムロジックを書ける力の方が重要ではないか。

- 前期はサーバーサイドJava。素のServlet + JDBCで泥臭くCRUDを書かせる。Spring Boot等のフレームワークは使わない。PHPで$_GETや$_POSTで受けていたものがJavaだとどう書くか──この対比で静的型付け、クラス構造、コンパイルの概念が肉体化される。フレームワークの便利さは後期でAIに書かせたときに実感すればよい。就職対応、C言語への橋渡しも確保
- 後期は自然言語設計。自然言語で要件定義書・設計書を書き、AIコーディングツール（Cursor, Claude Code等）で実装させ、出力されたコードを読解・検証・修正する
- 後期には Docker / GitHub Actions / AWSデプロイも統合し、クラウドインフラの入口とする

この案の設計思想：

- 1年で「自分の手でコードを書く」体験を十分に積んでいるからこそ、2年後期で「AIに書かせる側に回る」ことに意味がある
- Javaのボイラープレートの苦労を前期に体験しているからこそ、後期でAIにJavaを書かせたときに「ここがおまじないで、ここが本質」が分かる
- 「設計書類を残す力」が実戦的に鍛えられる。自然言語の精度がAIの出力品質を直接決めるため、教養OSで鍛えた分解力・構造化力がダイレクトに効く

出口職種との対応：

- 前期Java → クラウドインフラ（サーバーサイド理解）、IoT（C言語への橋渡し）
- 後期自然言語設計 → AI活用伴走（情シス拡張）の仕事そのもの
- Docker / GitHub Actions → クラウドインフラ（DevOps入口）

各案の比較：

観点	A: JavaScript	B: Java	C: Java前期+自然言語設計後期
AI協働学習	良好	工夫が要る	後期で本格的に実践
概念の肉体化	クライアント/サーバー統合	型・設計の厳密さ	型の肉体化 + 設計力
就職担当の納得	やや弱い	強い	Java経験ありと言える
3年への接続（組み込み）	距離がやや大きい	Cへの橋渡しになる	前期Javaで橋渡し確保
AWS授業との連携	フロントで繋がる	サーバーサイドで繋がる	後期でDocker/CI込みで深く繋がる
AI時代の先見性	中	低	高
設計書を書く力	弱い	弱い	後期で集中的に鍛える

4-3. 3年：2パス構造（Aコース / Bコース）

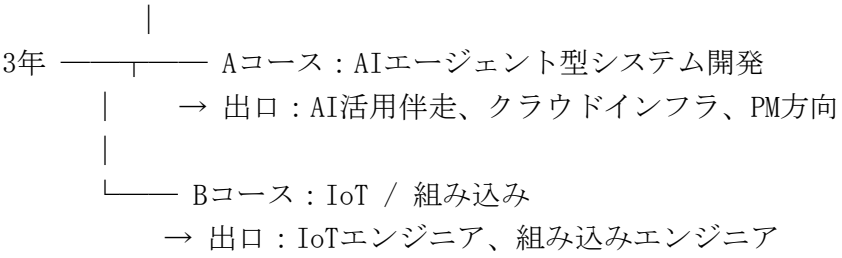
3年目は、学生の適性と志向に応じて2つのパスに分岐する。これは単なるコース選択ではなく、AI時代における**生存戦略としての分岐**である。生成AIがコーディングの大部分を代替する時代に、IT人材が生き延びる経路は大きく2つに分かれる：抽象的な設計業務―要件の整理、エージェントの管理、品質の最終判断―に関わっていく道と、物理世界との接点を持つ実装ベースの仕事―AIにはできないハードウェア制御、センサー統合、現場の泥臭い作業―で価値を出す道である。1年〜2年の共通基盤の上に、それぞれの生存経路に向けた専門性を積む。

この分岐の前提にある構造的な意味：

1年で「自分の手でコードを書く」、2年前期で「別言語で再体験する」、2年後期で「AIに書かせる側に回る」―ここまでの2年間は、実はAIエージェントの内部処理を自分で体験していたことになる。PHPでCRUDを書いた体験はコーディングエージェントの内部そのものであり、設計書を書いてAIに渡した体験はエージェントへの指示出しの練習だった。この2年間の蓄積があるからこそ、3年目でエージェントを「使う側」に立てる。

1年：PHP + Linux + 教養OS（全員共通）

2年：Java前期 + 自然言語設計後期（全員共通）



Aコース：AIエージェント型システム開発（設計型）

複数のAIエージェントに役割を分担させ、チームとしてシステムを開発する力を育てる。2年後期で「AI 1体に自然言語で設計→実装→検証」まで到達した力を、「複数体の編成・統合・品質保証」へ拡張する。

RPGのパーティー編成のたとえが分かりやすい：軍師（要件定義）、侍（コーディング）、防御型（レビュー）、斥候（テスト）、吟遊詩人（ドキュメント）。学生は監督（PM/プロデューサー）として、目標を与え、役割を割り当て、成果を統合し、品質を最終判断する。

Aコースの学生像：抽象的な設計が好き。業務を整理するのが得意。人やエージェントに指示を出して全体を動かす側。

設計原則：ツール非依存 × 体験先行（帰納）

AIエージェントの開発ツール（Devin、Claude Code、マルチエージェントフレームワーク等）は急速に変化しており、具体的なツール選定は開講時点の状況を見て判断する。教えるべきは特定ツールの操作ではなく、ツールが変わっても残る**原理原則**―役割設計、指示設計、統合、品質保証―である。授業は毎回「まず動かす→驚く→概念化」の帰納的学習サイクルで回す。

前期15回の概要：

回	テーマ	到達目標
1	エージェント・パーティーの最小実装（1→2体）	「役割2つで成果が増える/減る」を体感
2	役割設計の基本（責務境界と入出力）	各エージェントのI/O契約を定義できる
3	指示の精度管理（"仕様"として書く）	制約・例・禁止事項・品質条件に分解できる

4	ツール呼び出しと権限設計	ツールを使わせる/使わせないを設計できる
5	停止条件・再試行・失敗時ハンドリング	エージェント暴走を止める設計ができる
6	共有メモリと引継ぎ（コンテキストの分割統治）	コンテキストを共有/専用/一時の3層に分けられる
7	統合と矛盾検出（仕様×実装×テストの三角測量）	複数出力の矛盾をテストで潰せる
8	評価（Evals）の基礎	LLM出力を採点基準に落とせる
9	テストエージェント（斥候）の作り方	テスト生成→実行→レポートの流れを組める
10	セキュリティ監査エージェント（防御型の強化）	脆弱性観点をチェックリスト化して回せる
11	ドキュメントエージェント（吟遊詩人）と仕様の固定	変更とドキュメントを同期させられる
12	ワークフロー設計（グラフとしての″見える化″）	エージェント間協調をグラフで設計できる
13	業務エージェント化（RAG/文書フローの統合）	社内知識をエージェントに安全に組み込める
14	ミニプロジェクト（チーム制作）	役割設計→指示→統合→評価→改善を一周
15	発表＋運用設計レビュー	何を自動化し、どこを人が判断し、どう監査するか説明できる

後期：総合演習（エージェントチームでの本格開発プロジェクト）

評価の4軸（ツールが変わっても残る採点基準）：

1. **役割設計の明確さ**（責務境界とI/Oが定義されているか）
2. **指示の検証可能性**（例・制約・品質条件があるか）
3. **統合と矛盾解消の手順**（テストで決着しているか）
4. **運用・監査の再現性**（ログと停止条件があるか）

教材方針：

既存教材はフレームワーク別（LangGraph / CrewAI / AutoGen等）が多く、「ツールが変わっても残る運用・品質・統合の型」が弱い。独自教材として「型（テンプレート）集」を開発する。これはエージェント役割設計書、指示書テンプレート、受け渡し仕様、統合チェックリスト、レビュー観点表、トラブルシューティング・プレイブック、採点ルーブリックで構成され、学校の知的資産となる。

参考リソース：Microsoft "AI Agents for Beginners"（GitHub）、LangChain Academy（LangGraph）、DeepLearning.AI（crewAI / LangGraph）、OpenAI Cookbook（Agents）、Anthropic（context engineering）等。

→ 詳細設計（各回の演習内容・参考教材・独自教材の目次案）は別紙「3年Aコース詳細設計書」を参照。

Bコース：IoT / 組み込み（実装型）

物理世界との接点を持つシステムを、自分の手で作る力を育てる。

- 前期：組み込み開発（Arduino / Raspberry Pi / C言語）
  - センサー、デバイス、電子回路の基礎
  - C言語によるハードウェア制御
  - センサー → Bluetooth → クラウド可視化（ハードからクラウドまでの全体像）
  - 2年で学んだAI協働を組み込み開発にも適用（ソフト部分はAIに書かせ、ハード部分は自分の手で）
- 後期：総合演習（IoTシステムの本格開発プロジェクト）

Bコースの学生像：モノを動かすのが好き。手を動かして目に見える結果を出したい。センサーが反応したりLEDが光ったりするのが嬉しい。

Bコースの強み：物理世界との接点はAIに代替されない。LLMはコードは書けるが基板にはんだ付けはできない。ハードウェア知識＋クラウド連携＋AI活用の組み合わせが希少価値を生む。既存リソース（教員、ハードウェア、コース運営実績）が活用できる。製造業DX語彙との接続、インド連携（自動車整備）との接続も可能。

RTOSについて：

RTOSは清風の学生が3年次に本格習得する対象としては重すぎる。工学系の基礎（電気回路理論、信号処理等）を体系的に積んでいない情報処理特化型の学校であることを踏まえると、半年でC言語の基礎からRTOSまで到達するのは現実的でない。一方で、IoTキャリアの延長線上で将来触れる可能性はある。HAL（Hardware Abstraction Layer）化とAI支援の進展により、工学基礎なしでもタスク設計レベルでRTOSと付き合える可能性は年々高まっている。

Bコースでは「RTOSという世界がある」ことへの導入的接触にとどめる（FreeRTOSのデモでタスク切り替えを目で見る程度）。将来必要になった際にAI支援で自力学習できるよう、入口だけ作っておく。RTOS教育の経験を持つ教員には、Bコース全体の監修や希望者向けの課外ゼミで力を発揮してもらう。

RTOS理解に向けた要素技術の学習優先度：

将来RTOSに踏み込む必要が生じた場合に備え、自力学習（AI支援前提）の道筋を示しておく。

優先度	要素技術	なぜ必要か	清風カリキュラムとの接点
★★★	C言語の構造体・ポインタ	RTOSのAPIは構造体とポインタで成り立つ。ここが分からないとサンプルコードが読めない	3年Bコース前期で基礎に触れる
★★★	タスク・スレッドの概念	RTOSの核心は「複数タスクの並行実行」。優先度付きスケジューリングの考え方	2年後期の自然言語設計でAIへのタスク分割を経験（抽象レベルでの接点）
★★☆	割り込み（interrupt）	センサーイベントの処理に不可欠。「いま何をしていても、この信号が来たら即座に反応する」という仕組み	3年Bコースでセンサー制御時に概念として触れる
★★☆	排他制御（セマフォ・ミューテックス）	複数タスクが同じリソースを取り合う問題の解決策。デッドロックの理解	概念としてはDB授業のトランザクション制御と同構造

★☆☆	メモリマップ・レジスタ操作	ハードウェアを直接制御する際に必要。HALを使えば抽象化されるが、トラブル時に必要	清風カリキュラムでは扱わない。AI支援での後追い学習が前提
★☆☆	タイマー・クロック管理	リアルタイム制約（何ミリ秒以内に応答する）の設計に必要	清風カリキュラムでは扱わない。AI支援での後追い学習が前提

★★★は3年Bコースの中で意識的に触れておく。★★☆は概念レベルで接点を作っておく。★☆☆は「こういうものがある」と地図を見せるにとどめ、必要になったときにAIと一緒に学ぶ。

両コースの共通点：

- 1年〜2年の共通基盤（PHP、Java、Linux、教養OS、自然言語設計）の上に立つ
- 3年後期はどちらも総合演習として成果物を作り発表する
- どちらのコースでも2年後期で学んだAI協働の力は使い続ける

言語配置の全体パターン（検討中）：

2年目の案A〜Cと、3年目の2パス構造の組み合わせ：

パターン	1年	2年	3年	特徴
①	PHP	JavaScript	A:エージェント開発 / B:IoT	フロント経験 + 分岐
②	PHP	Java	A:エージェント開発 / B:IoT	就職直結 + 分岐
③（推奨）	PHP	Java(前期)+自然言語設計(後期)	A:エージェント開発 / B:IoT	AI時代の設計者育成 + 分岐

パターン③が最も一貫した設計思想を持つ：1年で手書き→2年前期でJava再体験→2年後期でAI1体との協働→3年Aコースで複数エージェント管理、という階段が明確。Bコースに進む場合もJavaからC言語への橋渡しが効く。

4-4. Javaの位置づけ：半年で「型の肉体化」を達成する

パターン③では、Javaを2年前期の半年間で扱う。フルスタックJavaエンジニアを育てることが目的ではない。目的は以下の3つ：

1. **静的型付けの概念を肉体化させる**：PHPの動的型付けしか知らない学生に、「型を宣言しないと動かない」世界を体験させる
2. **C言語への橋渡し**：Bコース（IoT）に進む学生にとって、Javaの厳密さがC言語への心理的障壁を下げる
3. **就職対応**：「Java経験あり」と言える。半年でも、素のServlet + JDBCでCRUDを書いた経験は実質的

Javaは「死ぬ」ことはないが、「若者が学ぶべき言語」としてのポジションは下がっていく見込みである。

- Javaが強い領域は「既存の大規模業務システム」の維持・移行
- しかしこの仕事こそAIによる自動化の影響を一番受けやすい
- 新規開発でJavaを選ぶケースは減少傾向（TypeScript、Go、Rust、Pythonへ）
- AI協働の時代にJavaの冗長さは足かせになる

一方で、Javaから完全に手を引くことのリスクもある：

- 就職担当との摩擦（求人票にJavaと書いてある）
- SI企業の採用担当の反応

しかし、PHPとJavaScript/TypeScriptで3年間しっかり概念を肉体化させた学生なら、Javaの社内研修で脱落するわけがない。むしろ「あの学校の学生はキャッチアップが早い」という評価になる。

Javaをやめるのではなく、「表面的な即戦力を装うこと」をやめる。 基礎力の高い学生を育てた結果として、就職後のキャッチアップが早い人材になる、という設計である。

4-5. LLMとの相性（AIフレンドリーな言語で高く登る）

LLMの言語別の「使い勝手」は大きく異なる。

- **Python**：AIが出すコードが短く、1行ずつ「何をしているか」聞きやすい。対話的試行錯誤がしやすい。AI協働学習に最適
- **JavaScript**：Pythonほどではないが十分良好。フロントエンド中心なら特に
- **Java**：ボイラープレートが多く、「どこが本質でどこがおまじないか」が見えにくい。ただし「AIに書かせて読み解く」スタイルなら逆に学習に使える

AIフレンドリーな言語を先にマスターして、自分が行けるところまで高く行く。作れるものの幅を広げて自信をつける。この戦略は、学生の学力を最大限に伸ばすことに直結する。

学生の学力を最大限に伸ばすことの方が、表面的な即戦力を装うことよりも大事である。

第5部：概念の肉体化マップ

5-1. 最重要概念：データ構造の設計

データ構造の設計は、本カリキュラムにおいて最も重要な概念として位置づける。

3つのレベルを3年間で段階的に体験させる：

レベル1：データをどう持つか（1年前期）

PHPの連想配列で体験する。

```
// これは配列
$students = ["田中", "山田", "鈴木"];

// これは連想配列
$student = [
    "name" => "田中",
    "age" => 20,
    "grades" => [80, 90, 70]
];
```

「この情報をどういう入れ物に入れるか」を考える訓練の起点。CRUDをやるときも、**まず「どういうデータ構造にするか」を考えてからコードを書く**習慣をここで叩き込む。

レベル2：データをどう整理して保存するか（1年後期）

PHPからMySQLに繋いでCRUDを行う。テーブル設計、正規化、リレーション。

「1年前期に連想配列で持っていたデータを、今度はテーブルに入れるとどうなるか」―レベル1からレベル2への接続がここで起きる。

レベル3：データをどう動かすか（2年）

クライアントとサーバーの間でデータが行き来する。JSONの設計が問題になる。

```
{
    "student": {
        "name": "田中",
        "grades": [80, 90, 70]
```

```
    }
}
```

「PHPの連想配列として持っていたものが、JSONという形でクライアントとサーバーの間を行き来している」ーレベル1→2→3が一本の線で繋がる。PHPの連想配列、DBのテーブル、JSONのデータ構造が全部「同じものの異なる表現」だと分かる瞬間が、概念の肉体化である。

帰納的導入：

データ構造は理論から教えない。まず何も教えずに作らせる。「生徒の成績管理プログラムを作れ」とだけ言う。学生は思いつきで変数を並べて、動くけどぐちゃぐちゃになる。困る。次に少し違う課題を出す。また困る。何度か繰り返した後で、初めて「連想配列」「テーブル設計」を名前として与える。概念が後から来て、自分の体験にラベルが貼られる。

自然言語の分解訓練との統合：

プログラミング授業の中に「日本語タイム」を組み込む。例えば「ラーメン屋の注文管理」という題材で：

- AIと対話して「注文」という言葉を分解する → 誰が、いつ、何を頼んだか
- 「何を」をさらに分解する → メニュー名、サイズ、トッピング
- トッピングは1つ？複数？ → 1対多の関係
- この対話の結果を紙に書かせると、それがそのままER図の原型になる

5-2. その他の概念と肉体化の時期

概念	1年	2年	3年
CRUD	PHP（コマンドライン→Web）	2年言語で再体験	3年言語で再体験
クライアント/サーバー	PHP（後期で初体験）	JS/Java（両側体験）	クラウド・組み込みで統合
認証・認可	PHP（セッション認証）	トークン認証	フレームワークの認証機構
型（動的→静的）	PHP（動的）	JSまたはJava	TypeScriptまたはC

API	−	初体験（fetch AP　業務レベルのAPI設計I）
非同期処理	−	JS（fetchで初接　本格的理解触）

各概念は最低2回、異なる言語・文脈で体験させる。そのたびに「前回と今回の共通点と相違点」を言語化させる。これが概念の肉体化を促す。

第6部：横断的要素

6-1. セキュリティ（溶かし込み方式）

セキュリティは独立科目にしない。各学年のプログラミング授業に溶かし込む。

現状はクロスドメインエラーの回避程度で、セキュリティ観点は低い。「作らせるのが精一杯」が正直な現状である。しかしAI時代にはここが逆転する。AIが書くコードは「動く」がセキュリティホールがある可能性が高い。「守る視点」は人間が持っていないと誰も持っていない。

溶かし込みの具体策：

1年（PHP）：「壊す体験」を1〜2回

学生が作ったCRUDアプリに対して、教員が目の前でSQLインジェクションをやってみせる。データが全部消える。「えっ」という衝撃が帰納的学習の起点になる。そこから「なぜ壊れたのか」「どうすれば防げるのか」を考えさせる。

2年以降：毎回1分の「攻撃者視点タイム」

新しい機能を作るたびに「これを攻撃者ならどう悪用するか」を1分だけ考えさせる。独立した授業は不要。ただし毎回1分の習慣化がセキュリティマインドを育てる。

教養OSとの接続：

構造の型の「なぜか（因果）」「代わりに失うもの（代償）」はセキュリティの思考そのものである。「この対策をしないと何が起きるか」「この利便性を得る代わりに何を失うか」。

6-2. 設計書類を残す力

設計書類を書く力は、教養OSの4つの型が入っていれば自然に身につく：

- **構造の型**で要件を定義・比較・因果で整理する
- **描写の型**で現状を正確に言葉にする
- **修辞の型**で読み手に合わせて書く

独立科目にする必要はなく、プログラミング授業の中で「作る前に書く、作った後に残す」を習慣化する。

AI時代には設計書類の質がAIの出力の質を決める。設計書類を書く力は、実質的に**AIへの指示書を書く力**と同義である。

6-3. クラウド（AWS）との連携

現状、2年・3年で並行してAWSの授業を実施している。基本はXAMPPでやったPHPアプリをクラウドにデプロイすることが主目的。

クラウド授業はプログラミング言語の選択とは独立して動いており、言語授業との連携の余地がある：

- 2年でJavaScriptをやる場合：フロントのデプロイ、PHPサーバーへのAPI通信が自然に繋がる
- 2年でJavaをやる場合：サーバーサイドのデプロイとして繋がる
- 3年で組み込みをやる場合：センサーデータをクラウドで受けて可視化する全体像が完成する

第7部：出口職種と生成AIの影響分析

7-1. 想定される出口職種

就職担当との協議から、システム専攻の学生の主な出口として以下の3職種が挙がっている。

1. **業務アプリ開発**
2. **IoT関連**
3. **キッティング（IT運用）**

7-2. 各職種への生成AIの影響

業務アプリ開発：影響大（変質）

コーディング作業はAIに代替されるが、「何を作るか決める」「業務を理解してデータ設計する」「AIが出したコードを検証する」仕事は残る。本カリキュラムで重視する「概念の肉体化」「データ構造の設計力」「自然言語の分解力」が直接的に生存スキルとなる。「コーダーとして入る」なら危険、「業務を理解してAIと協働できる人材として入る」なら安全。

IoT関連：影響小〜中（追い風）

IoTは物理世界との接点があるため、AIに代替されにくい。センサー、デバイス、配線、電波、ハードウェアの制約はAIが触れない領域。ハードウェアや電子回路の知識が必要なことから新規参入のハードルが高く、人材の世代交代が進みにくいため、スキルを身につければ希少価値の高い人材となる。IoTとAI両方に対応できるエンジニアに需要が集中する見込みが高い。

キッティング（IT運用）：影響大（変質）

キッティングの仕事には2層ある。下層（作業の実行：箱を開けて設定する）はWindows Autopilot + Microsoft Intuneによるゼロタッチプロビジョニングでほぼ自動化される。上層（要件定義と設計書：何台に何のソフトを入れ、セキュリティポリシーをどう設定するか）はむしろ重要性が増す。「100回手を動かす仕事」から「1回正しく設計する仕事」への変質。キッティングだけで止まると先がないが、要件を整理して設計書に落とせれば、インフラエンジニアやクラウドエンジニアへのキャリアパスが開ける。

7-3. 生成AIが追い風となる出口分野

情報処理分野で、清風の学生が現実的に狙える「追い風」ポジションは以下の3つである。

クラウドインフラ / DevOps

AIはクラウドの上で動く。AIエージェントが増えるほど、インフラの設計・構築・運用が要る。AWS、Azure、GCPの設計・運用ができる人材は慢性的に不足。現在のAWS授業の延長線上にある。

IoT / 組み込み / エッジコンピューティング

物理世界との接点はAIに代替されない。むしろAIの「目」と「手」になるデバイスを作る側。ハード知識＋クラウド連携＋AI活用の組み合わせが希少価値を生む。3年の組み込みカリキュラムが直接ここに繋がる。

AI活用の伴走支援（情シス拡張）

「AIを導入したいが、やり方が分からない中小企業」が大量に出る。業務を理解して、適切なAIツールを選定・導入・運用する仕事。ビジネスOS検定で業務が分かり、プログラミングの概念が肉体化し、AIとの協働経験がある人材は、この仕事に最適。

3年間の階段構造と出口職種の対応：

年次	主軸	階段の意味	追い風分野への接続
1年（PHP + Linux + 教養OS）	自分の手でコードを書く	エージェント内部の自己体験	AI活用伴走の基礎
2年前期（Java）	別言語で再体験	型の肉体化、C言語への橋渡し	クラウドインフラの基礎
2年後期（自然言語設計）	AI 1体との協働	エージェントへの指示出し練習	AI活用伴走の実践
3年Aコース（エージェント開発）	AI複数体のチーム管理	PM/監督としてのシステム開発	→ AI活用伴走、クラウドインフラ
3年Bコース（IoT）	物理世界との接点	AIが入れない領域の実装力	→ IoT / 組み込み

これら全ての段階に共通して求められるのは「業務を理解する力」「設計する力」「AIと協働する力」であり、本カリキュラムの「陳腐化しない芯」がそのまま共通基盤となっている。

第8部：未決事項と次のステップ

> **注記：** 2年後期「自然言語設計」科目の詳細設計は別紙「2年後期 自然言語設計 詳細設計書」を参照。 3年Aコース「AIエージェント型システム開発」の詳細設計は別紙「3年Aコース 詳細設計書」を参照。

8-1. 未決事項

1. **2年目の言語選択**：案A〜Cの3案から選択。特に案C（Java前期＋自然言語設計後期）は教員の反応を踏まえて判断
2. **教養OSとプログラミング授業の時間配分**：日本語タイムをどの程度の頻度で入れるか
3. **教員体制**：新カリキュラムに対応できる教員の確保・育成。特に案Cの後期（AI協働設計）を教えられる教員

4. **就職担当との合意形成**：出口職種と追い風分野の分析を共有し、「即戦力」の定義を更新する
5. **独自テンプレート集の詳細設計**：教養OSの型の詳細設計と同時進行で完成させる必要がある
6. **AIツール選定**：Claude Code / Cursor / Copilot のうち、学生向けライセンスと授業運用に最適なものを開講時点で判断
7. **赤俊哉セオリーシリーズの授業接続ポイントの特定**：購入前に、どの章が本科目のどの回に対応するかを精査する

8-2. 次のステップ

1. **教員への案C提示と反応の収集**：Java前期＋自然言語設計後期の案に対するフィードバック
2. **出口職種と追い風分野の分析を就職担当と共有**：キッティングの変質、クラウドインフラ/IoT/AI伴走の追い風を共有
3. **教養OSの1年前期モジュール（構造の型7回 + 描写の型7回）をプログラミング授業との連携で詳細設計する**
4. **ビジネスOS検定のユニット1〜3の語彙リストと、プログラミング題材の対応表を作成する**
5. **AI壁打ち用のプロンプトテンプレート（自然言語分解訓練用）を設計する**
6. **独自テンプレート集のドラフト作成**：要件定義書・データ設計書・画面設計書・AIへの指示書・チェックリストの初版を作成
7. **赤俊哉セオリーシリーズの読了と授業接続マッピング**
8. **ゲーム専攻・広告デザイン専攻への展開案を作成する**

本文書は2025年2月20日時点の構想を記録したものであり、今後の議論・検証を経て更新される。

2年後期「自然言語設計」科目　詳細設計書

2年後期「自然言語設計」科目 詳細設計書

作成日： 2025年2月20日 **位置づけ：** カリキュラム再設計構想書の別紙。2年後期・案C採用時の詳細設計。

---

1. 科目の位置づけと設計思想

本科目は教養OSの「実技編」である。

教養OSで鍛えた4つの型（構造・描写・物語・修辞）を、実際にシステムの設計書を書き、AIに実装させ、検証・修正するという実務サイクルに落とし込む科目である。

バイブコーディングとの明確な区別：

「バイブコーディング」（vibe coding）とは、設計なしに雰囲気でAIにコードを書かせる行為を指す。自然言語設計はその対極に位置する。設計を精密にした上でAIに渡すからこそ、出力の質が上がり、検証も可能になる。「設計の精度がAI出力の品質を決める」──これが本科目の核心命題である。

前提となる学生のスキル（2年後期開始時点）：

- PHP（1年通年）：CRUD、セッション認証、DB接続、MVC概念を自分の手で実装した経験
- Java（2年前期）：素のServlet + JDBCでCRUDを書いた経験。静的型付け、クラス構造、コンパイルの概念
- Linux基礎（1年）：bash、パイプ、リダイレクト
- 教養OS（1年通年）：構造の型、描写の型、物語の型、修辞の型
- AWS授業が並走中

2. 教材戦略：ハイブリッド方式

既存の教科書1冊で完結する科目ではない。以下のハイブリッド方式を採る。

主教材（学生購入）：

- 赤俊哉『だまし絵を描かないための――要件定義のセオリー』（リックテレコム）：要件定義テンプレの骨格、用語・粒度・合意の取り方
- 赤俊哉『ユーザー要求を正しく実装へつなぐ システム設計のセオリー』（リックテレコム）：設計成果物の観点辞書

この2冊は「自然言語で設計を書く」ときの観点と品質基準を提供する。ただし、これらは本科目の「背骨」であって「全体」ではない。

Web教材（毎回配布・リンク）：

AIツールとインフラツールは更新が速いため、紙の教科書ではなく公式ドキュメントを教材化する。Claude Code公式ドキュメント、Docker Get Started（日本語版）、GitHub Actions公式日本語ドキュメント、GitHub Education / Cursor 学生向けプログラム等。

独自教材（本科目の"教科書"本体）：テンプレート集＋チェックリスト

教養OSの型と直接接続したテンプレート群。詳細は8-3参照。

教員参照（図書室）：

- 吉原庄三郎『はじめての設計をやり抜くための本 第2版』（翔泳社）

3. 独自テンプレート集の設計

本科目の教材の核心は、教養OSの型をシステム設計の成果物に変換するテンプレート群である。学生は毎回「AIに渡せる中間成果物」をこのテンプレートに沿って書き、その精度でAI出力の質が変わることを体験する。

テンプレート1：要件定義書（教養OS：構造の型ベース）

欄	教養OSの型との対応	書く内容
目的	構造の型「定義」	これは何のためのシステムか（非ITの言葉→ITの言葉）
利用者と利用状況	描写の型	誰が・いつ・どこで・何を困っているか
現状業務フロー	描写の型→構造の型「因果」	現状の手順と、詰まりの原因
スコープ	構造の型「比較」「代償」	やること／やらないこと、代わりに何を失うか
機能要求	構造の型「定義」	ID付き（FR-001…）＋受入条件

非機能要求	構造の型「代償」	性能・可用性・セキュリティ・監査ログ
受入テスト観点	構造の型「定義」	ユーザーがOKと言う条件

テンプレート2：データ設計書（自然言語でテーブル設計）

- エンティティ定義：**「何を1行とするか」**を日本語で明示（例：生徒1人＝Student 1行）
- 属性一覧：意味（定義）／型／必須／例／バリデーション
- 関連：1対多・多対多を日本語で記述
- 制約：一意・参照整合・削除時の挙動
- サンプルデータ（5行）：AI生成結果の検証用

1年のDB授業でテーブル設計をER図で書いた経験を、**あえて自然言語に翻訳し直す**訓練。
「コードで書けることを、自然言語で書き直す」逆変換こそが本科目の本質。

テンプレート3：画面設計書（教養OS：物語の型ベース）

- シナリオ（起→承→転→結）：ユーザーの期待→障害→解決→達成
- 画面一覧（UI-001…）：目的／入力／表示／操作／エラー
- 画面遷移：条件付き遷移を日本語で記述
- 文言設計：留学生対応（やさしい日本語）版の文言欄

物語の型の「期待→裏切り→回収」構造が、そのままユーザーシナリオの「正常系→エラー→リカバリ」に対応する。

テンプレート4：AIへの指示書（プロンプトの構造化）

- 目的：何を作るか
- 制約：技術スタック（PHP/Java/DB等）・禁止事項
- 入力：要件定義書・データ設計書・画面設計書を貼る欄
- 出力形式：ファイル構成、コードブロック規約、コメント規約
- 検証手順：テスト方法、起動方法、想定入力
- 失敗時の再指示：エラーを貼る欄＋「どこを直すか」の問い

テンプレート5：コードレビュー・検証チェックリスト

- セキュリティ：認可漏れ、SQLインジェクション、XSS、CSRF、秘密情報の露出
- ロジック：境界値、状態遷移、重複、例外時の整合性
- DB：N+1、インデックス、トランザクション境界
- 運用：ログ出力、設定の外出し、再現手順

- AI特有：**仕様にない機能が勝手に混入していないか**（自然言語起点の事故として最も多い）

4. 15回カリキュラム（半期）

全体を貫くプロジェクト題材は**「小さな業務アプリ」**（例：校内備品貸出、補講申請、出欠管理＋簡易通知など）。ビジネスOS検定の語彙圏で、1年のCRUD経験と直結する題材を選ぶ。

原則：座学は最小限。毎回「手を動かす」「成果物が出る」設計にする。

■ 第1ブロック（第1〜7回）：設計力―自然言語で「実装可能な精度」にする

第1回：科目導入―AIに「雑に」書かせて失敗する体験

- 学習目標：AIの得意不得意を体感し、「設計の精度がAI出力の品質を決める」を実感する
- 内容：曖昧な1行指示でAIにアプリを作らせる → 動くが仕様と違う・セキュリティ穴だらけ → 同じ機能を精密な指示で作り直す → 品質の差を比較する
- 成果物：失敗プロンプトと改善プロンプトの比較記録（AIへの指示書テンプレの最小版）
- ツール：Claude Code または Cursor
- 教養OSとの接続：構造の型「比較」（2つの指示の何が違って何が変わったか）

第2回：要件定義（1）―現状業務の描写と目的の定義

- 学習目標：描写の型で現状業務を言語化し、構造の型で目的を定義する
- 内容：プロジェクト題材の業務を「主語・動詞・対象」の3点セットで記述する訓練。AIとの壁打ちで「それは具体的に何？」を繰り返し、分解精度を上げる
- 成果物：要件定義書テンプレート（目的・利用者・現状業務フロー）
- 教養OSとの接続：描写の型（観察→言語化）、構造の型「定義」

第3回：要件定義（2）―スコープと受入条件

- 学習目標：「やること／やらないこと」を決める判断力。代償の型で「何を捨てるか」を言語化する
- 内容：機能要求をID付きで列挙。各機能に受入条件（Given/When/Then形式）を書く
- 成果物：要件定義書テンプレート（スコープ・FR一覧・受入条件）
- 教養OSとの接続：構造の型「比較」（やる／やらない）、「代償」（捨てるものの明示）

第4回：設計（1）―データ設計（自然言語でテーブルを書く）

- 学習目標：1年のDB経験を「あえて自然言語に翻訳し直す」ことで、設計の本質を再認識する
- 内容：エンティティ定義・属性・関連・制約をすべて日本語で書く。ER図は「結果として出てくるもの」であって、まず日本語が先
- 成果物：データ設計書テンプレート
- 教養OSとの接続：構造の型「定義」（何を1行とするか）、描写の型（属性の正確な記述）

第5回：設計（2）―画面設計（物語→画面）

- 学習目標：ユーザーシナリオを物語の型で書き、そこから画面を導出する
- 内容：「ユーザーが〇〇しようとして→△△が起きて→□□で解決する」をシナリオとして書く。シナリオから画面一覧と遷移を抽出する
- 成果物：画面設計書テンプレート（シナリオ・画面一覧・遷移・文言）
- 教養OSとの接続：物語の型「期待→裏切り→回収」= 正常系→エラー→リカバリ

第6回：設計（3）―処理フローの自然言語化（コード→日本語の逆変換）

- 学習目標：自分が前期Javaで書いた処理を、AIに渡す精度の日本語に翻訳する
- 内容：前期に書いたServlet + JDBCのCRUD処理を題材に、「このコードは何をしているか」を自然言語で再記述する。分岐・例外・状態遷移を日本語で書き切る訓練
- 成果物：主要ユースケースの処理フロー記述（自然言語版）
- 教養OSとの接続：構造の型「因果」（このデータが来たら→この処理が走る→この結果が返る）
- 注記：従来の「疑似コード→コード」ではなく「コード→自然言語」の逆変換。1年半コードを書いてきた学生だからこそ意味がある訓練

第7回：設計レビュー―設計書の「穴」を見つける訓練

- 学習目標：他者の設計書を検証する目を養う。チェックリストの使い方を体得する
- 内容：ペアまたはグループで設計書を交換し、チェックリストに沿ってレビュー
- 成果物：レビュー指摘票（チェックリストベース）
- 教養OSとの接続：構造の型「因果」「代償」（この設計だと何が起きるか、何を見落としているか）

■ 第2ブロック（第8〜11回）：AI協働開発―生成→検証→修正のサイクル

第8回：AI実装（1）―設計書をAIに渡して雛形を生成する

- 学習目標：AIへの指示書テンプレートを使い、設計書からコードを生成させる
- 内容：第2〜6回で作った設計書をAIへの指示書に組み込み、プロジェクトの初期コードを生成。ファイル構成、README、起動手順まで含めてAIに出させる
- 成果物：生成されたコード一式＋起動手順（README）
- ツール：Claude Code（複数ファイル編集・コマンド実行の特性が授業向き）

第9回：AI実装（2）―受入条件をテストに落とす

- 学習目標：第3回で書いた受入条件を、テストケースとして具体化する
- 内容：Given/When/Then形式の受入条件をテストケースに変換。可能な範囲で自動テスト化。AIにテストコードも書かせ、その妥当性を検証する
- 成果物：テストケース一覧＋自動テスト（可能な範囲で）

第10回：AI実装（3）―バグ修正と仕様逸脱の検知

- 学習目標：AIの出力を「疑う」技術。仕様にない機能の混入を見抜く
- 内容：意図的にバグを含むAI出力を渡し、発見・報告・再指示のサイクルを回す。プロンプト履歴を記録し、「どの指示で改善したか」を追跡する
- 成果物：バグ報告書＋再指示ログ（プロンプト履歴）
- AI特有の落とし穴：仕様書にない機能を「良かれと思って」AIが追加するパターンへの対処

第11回：セキュリティと品質―AIが書いたコードの最低限の防波堤

- 学習目標：検証チェックリストを使い、AIが書いたコードのセキュリティ・品質を検証する
- 内容：認可漏れ、SQLi、XSS、CSRF、秘密情報の露出をチェックリストに沿って検査。発見した問題をAIに修正させ、修正結果を再検証する
- 成果物：脆弱性チェック結果＋修正記録
- 教養OSとの接続：構造の型「因果」「代償」（この脆弱性を放置するとどうなるか）

■ 第3ブロック（第12〜14回）：インフラ力―Docker・CI/CD・デプロイ

第12回：Docker化―「誰でも同じ環境で動く」を作る

- 学習目標：開発環境をコードで定義する概念を理解する（これも「自然言語設計」の延長）
- 内容：プロジェクトのDockerfile + docker-compose.ymlを作成
- 成果物：Dockerfile + compose + 起動手順書
- 1年Linuxとの接続：Dockerの中はLinux。1年で覚えたbashコマンドがそのまま使える

第13回：GitHub Actions―pushしたらテストが自動で回る

- 学習目標：CI（継続的インテグレーション）の概念。1年のパイプの概念との再会
- 内容：GitHub Actionsのワークフローを書き、pushのたびにlintとテストが自動実行される仕組みを構築
- 成果物：.github/workflows/ci.yml（動くもの）
- 1年Linuxとの接続：GitHub Actionsの中身はbashスクリプトのパイプライン

第14回：AWSデプロイ―作ったものをクラウドに載せる

- 学習目標：並走するAWS授業との接続点を作る
- 内容：Docker化したアプリをAWSにデプロイする最小限の手順を体験。Secrets管理の基礎
- 成果物：デプロイ手順書＋Secrets運用メモ
- 注記：AWS授業の到達点に合わせて深さを調整

■ 第4ブロック（第15回）：統合―全サイクル発表

第15回：最終発表と振り返り

- 学習目標：設計→AI生成→検証→インフラの全サイクルを自分の言葉で説明できる
- 提出物（必須）：要件定義書／データ設計書／画面設計書／AIへの指示書（最終版）、生成ログ（主要3回分のプロンプト履歴）、Docker + CI（動作確認済み）、振り返りレポート：因果分析（構造の型で「なぜうまくいった／いかなかった」を分析）
- 評価基準：コードの出来ではなく、設計書の精度・検証の深さ・振り返りの質

5. 15回の全体構造図

第1ブロック：設計力（第1〜7回）
　｜　教養OSの型 → 設計書テンプレート → 「AIに渡せる精度」の成果物
　↓
第2ブロック：AI協働開発（第8〜11回）
　｜　設計書 → AI生成 → 検証 → 修正 → 再生成のサイクル
　↓
第3ブロック：インフラ力（第12〜14回）
　｜　Docker → CI/CD → AWSデプロイ（1年Linuxとの再会）
　↓
第4ブロック：統合（第15回）
　｜　全サイクルの発表 → 振り返り → 3年への橋渡し

この15回で達成されること：

- 教養OSの4つの型が「設計書を書く実務力」として肉体化される
- 「AIにコードを書かせる」体験を通じて、設計の精度と出力品質の因果関係を実感する
- Docker / GitHub Actions / AWSという開発インフラの基礎が入る
- 3年Aコース（エージェント開発）への直接的な準備が完了する
- 3年Bコース（IoT）に進む学生も、ソフト部分のAI協働力を持ってBコースに入れる

3年Aコース「AIエージェント型システム開発」詳細設計書

3年Aコース「AIエージェント型システム開発」詳細設計書

作成日： 2025年2月20日 **位置づけ：** カリキュラム再設計構想書の別紙。3年Aコース前期15回の詳細設計。 **前提：** 2年後期で「AI 1体に自然言語で設計→実装→検証」まで到達している学生を対象に、「複数体の編成・統合・品質保証」へ拡張する。

---

1. 設計原則

- **ツール非依存：** 特定フレームワーク（LangGraph / CrewAI / AutoGen等）の操作を教えるのではなく、ツールが変わっても残る原理原則を教える
- **体験先行（帰納）：** 毎回「まず動かす→驚く→概念化」のサイクル
- **2年後期からの接続：** 自然言語設計で培った指示書作法を、役割別に分割・拡張する形

---

2. 既存教材・参考リソース

A. 授業でそのまま使える実践系

リソース	位置づけ	学生適合
Microsoft "AI Agents for Beginners"（GitHub・無料）	エージェント基礎〜実装の入り口。レッスン形式で授業回に分解しやすい	◎
LangChain Academy：Intro to LangGraph（無料）	「ワークフロー＝グラフ」で複数エージェント協調を見える化	◎
DeepLearning.AI：Multi AI Agent Systems with crewAI	「役割分担」「チームで業務プロセスを回す」に直結	◎

DeepLearning.AI：AI Agents in LangGraph	スクラッチ→フレームワークで再構成の流れが帰納学習に合う	◎
AutoGen関連教材（DataCamp記事等）	複数エージェント会話・役割・ツール接続の典型例	○（環境差分が出やすいため教員側で型を固定）

B. 設計・品質保証まで伸ばす公式ドキュメント系（教員の背骨）

リソース	授業での用途
OpenAI Cookbook：Agents / Function calling	ツール呼び出し、ガードレール、停止条件、観測可能性の原理原則
Anthropic：Prompting best practices / Context engineering / Writing tools for agents	指示の精度管理の体系化。コンテキスト設計、ツール仕様の書き方
LangChain Docs：Agents / Workflows vs Agents	「ワークフロー（決め打ち）」と「エージェント（動的）」の差を腹落ちさせる
LlamaIndex：Agent Workflows / Agentic Document Workflows	複数エージェントのオーケストレーションをワークフローとして整理

C. 理論の裏取り・学術系（教員の引き出し、学生に通読させない）

- Shoham & Leyton-Brown：*Multiagent Systems*（基礎教科書）
- Wooldridge：*An Introduction to MultiAgent Systems*

使い方：「役割」「利害/情報の非対称」「協調と競合」「プロトコル」などを概念ラベルとして後から貼る用途。

D. プロンプト工学の体系（指示書作法の共通基盤）

- Prompt Engineering Guide（Web）：辞書的参照

- DeepLearning.AI：ChatGPT Prompt Engineering for Developers：演習向き

---

3. 先行事例と取り入れる要素

大学・教育機関

Stanford "CS146S: The Modern Software Developer"

- 現代の開発者像（AIネイティブ、DevOps、オンコール等）にマルチエージェントの話題が組み込まれている
- 取り入れる要素：
  - "作る"だけでなく運用（障害対応・改善）を含む評価軸
  - 役割分担＝「開発」「レビュー」「オンコール」「ドキュメント」に広げる

実務寄りオンライン教育

- LangChain Academy（LangGraph）
- DeepLearning.AI（crewAI / LangGraph）
- Microsoft GitHub教材（AI Agents for Beginners）

企業研修

- NobleProg：AutoGenトレーニング（企業向け講座として成立＝トピック分解が参考になる）

先行事例から抽出した授業設計の要点

1. **フレーム（RPG役割）を固定して混乱を減らす**
2. 出来た/出来ないは**成果物ではなくプロセス品質**（指示書・引継ぎ・テスト・統合）で採点
3. **観測可能性（ログ/差分/再現）を最初から必須要件にする**（後からだと崩壊する）

---

4. 半期15回カリキュラム詳細

毎回の授業フォーマット（共通の型）

- **導入（10〜20分）**： 今日の"パーティー編成"と勝利条件（Definition of Done）
- **演習（60〜120分）**： 小さく回す（最小のループ→改善）

- **振り返り（20〜30分）：** うまくいった指示／ダメだった指示を比較。"概念ラベル"付け（例：停止条件、責務境界、コンテキスト設計、評価指標）

第1回：エージェント・パーティーの最小実装（1→2体）

- **到達目標：** 「役割2つで成果が増える/減る」を体感する
- **演習：**
  - コーディング役＋レビュア役の2体で、既知の小課題（CRUD APIの一部など）を改修
  - 2年後期で使った「AI 1体」方式と差分比較
- **ツール例：** Cursor/Claude Code等のIDE支援＋任意の簡易フレーム（LangGraph/CrewAI/AutoGenどれでも）
- **接続点：** 自然言語設計の「指示書」を役割別に分割する

第2回：役割設計の基本（責務境界と入出力）

- **到達目標：** 各エージェントの入力/出力（契約）を定義できる
- **演習：** RPG 5役（軍師/侍/防御/斥候/吟遊詩人）のI/Oテンプレに落とす

第3回：指示の精度管理（"仕様"として書く）

- **到達目標：** 曖昧指示を、制約・例・禁止事項・品質条件に分解できる
- **参考：** Anthropic Prompting best practices / context engineering

第4回：ツール呼び出し（tool/function calling）と"権限設計"

- **到達目標：** ツールを"使わせる/使わせない"を設計できる
- **演習：** 「DB参照は可、テーブル変更は不可」などの権限境界を設計
- **参考：** OpenAI function calling
- **接続点：** 2年後期のDocker/GitHub Actions/AWSを「道具」として再利用

第5回：停止条件・再試行・失敗時ハンドリング

- **到達目標：** エージェント暴走（無限ループ/無駄作業）を止める設計ができる
- **演習：** タイムアウト、最大ステップ、失敗時の人間エスカレーション

第6回：共有メモリと引継ぎ（コンテキストの分割統治）

- **到達目標：** コンテキストを「共有」「役割専用」「一時」の3層に分けられる
- **参考：** Anthropicの"context engineering"

第7回：統合（merge）と矛盾検出（仕様/実装/テストの三角測量）

- **到達目標：** 複数出力の矛盾を、テスト・仕様・ログで潰せる
- **演習：** 同じAPIに対する複数実装案を統合してテストで決着

第8回：評価（Evals）の基礎：LLM出力を採点可能にする

- **到達目標：** ″良い/悪い″を、採点基準に落とす
- **演習：** 期待JSONに一致、禁止語なし、手順遵守、等の自動判定
- **参考：** OpenAI Cookbook（agents系）

第9回：テストエージェント（斥候）の作り方

- **到達目標：** テスト生成→実行→レポートまでの流れを組める
- **演習：** pytest等で、失敗→原因切り分け→指示修正ループ

第10回：セキュリティ監査エージェント（防御型の強化）

- **到達目標：** 脆弱性観点を″チェックリスト化″して回せる
- **演習：** AIのコードに対し、静的解析・依存脆弱性・危険API使用を点検（Semgrep/Bandit等は例）

第11回：ドキュメントエージェント（吟遊詩人）と「仕様の固定」

- **到達目標：** README/API仕様/運用手順を、変更と同期させられる
- **演習：** PRテンプレ＋CHANGELOGをエージェントに更新させ、レビューエージェントが整合チェック

第12回：ワークフロー設計（LangGraph的な″見える化″）

- **到達目標：** エージェント間協調をグラフとして設計できる
- **参考：** LangChain Academy / LangGraph Docs

第13回：ドキュメント／社内知識を使う″業務エージェント″化

- **到達目標：** RAG/文書フローを、エージェントに組み込む際の事故を減らす
- **参考：** LlamaIndex Agentic Document Workflows

第14回：ミニプロジェクト（チーム制作）– パーティー運用

- **到達目標：** 役割設計→指示→統合→評価→改善を一周
- **演習：** 小さな業務アプリ（例：問い合わせ分類＋返信草案＋チケット化等）を、複数エージェントで完成

第15回：発表＋″運用設計レビュー″（作って終わりにしない）

- **到達目標：** 何を自動化し、どこを人が判断し、どう監査するかを説明できる
- **成果物：**
    - エージェント構成図（グラフ）
    - 指示書一式
    - 評価（Evals）
    - 監査ログ方針
- **先行事例の要素：** 運用まで含めた評価（Stanford型）

---

5. 評価設計

評価の4軸（ツールが変わっても残る採点基準）

1. **役割設計の明確さ**（責務境界とI/Oが定義されているか）
2. **指示の検証可能性**（例・制約・品質条件があるか）
3. **統合と矛盾解消の手順**（テストで決着しているか）
4. **運用・監査の再現性**（ログと停止条件があるか）

成果物の派手さではなく、プロセス品質を点数化する。この4軸は就職先でそのまま効く。

---

6. 独自教材（学校の知的資産）構成案

コンセプト

- タイトル案：**『AIエージェント・パーティー開発：役割設計／指示設計／統合／品質保証の型』**
- 価値：企業が一番困るのは「導入」より「運用で壊れる」ところ。ここをテンプレ化すると就職に直結する差別化になる

目次案（テンプレート駆動）

章	テーマ
1章	エージェント開発の全体像（人間1→AI1→AI複数）
2章	役割設計（責務境界／I/O契約／禁止事項）

3章　指示書の書き方（制約・例・品質条件・失敗時）

4章　協調設計（Aの出力をBの入力にする"受け渡し仕様"）

5章　停止条件・再試行・エスカレーション設計

6章　統合（merge）と矛盾検出（仕様×実装×テスト）

7章　評価（Evals）– 採点可能にする技術

8章　セキュリティ監査の型（レビュー観点の固定）

9章　ドキュメント同期の型（README/仕様/運用手順）

10章　運用と監査（ログ、再現性、責任分界）

付録　ケーススタディ（学校向け業務題材 6本）

テンプレート集（この授業の"勝ち筋"）

- **エージェント役割設計書（1枚）：** 目的 / 入力 / 出力 / 使って良いツール / 禁止 / 品質条件 / 停止条件
- **指示書テンプレート（詳細版・短縮版）**
- **受け渡し仕様（Handoff Spec）：** A→Bに渡す形式（JSON/Markdown/表）
- **統合チェックリスト：** 仕様・テスト・ログ・依存関係
- **レビュー観点表：** "おまじない vs 本質"判定を含む
- **トラブルシューティング・プレイブック：** 期待と違う出力の原因を「指示」「コンテキスト」「ツール」「停止条件」「評価」に切り分ける手順
- **採点ルーブリック：** 成果物より"プロセス品質"を点数化（役割の明確さ、指示の検証可能性、統合の再現性、評価の客観性）

---

7. 次のステップ

1. 開講時点のツール動向を見て、演習で使うフレームワーク（LangGraph / CrewAI / AutoGen等）を1〜2に絞る
2. テンプレート集のプロトタイプを作成し、教員レビュー

3. 第1〜4回の配布プリント＋演習課題＋採点表を先行作成してパイロット実施
4. 演習で使う言語（Python / TypeScript / Java）を、2年後期の制作スタックとの接続で決定

GoStop設計人材とシリコンバレー東京バンガロール三極連携 — 成功体験量産と評価設計

GoStop設計人材とシリコンバレー東京バンガロール三極連携 – 成功体験量産と評価設計

【論点】

1. 偏差値40層への教育設計はどうあるべきか – 成功体験の「量産」
2. 成功体験が少ない人に、どうすれば「挑戦」を教えられるか
3. 評価設計を変えることで、なぜ教育が変わるか
4. グラウンディング（体験重視）と抽象論の順序

【対比】

成功体験が少ない人の状態	Go/Stop設計後の状態
どこから始めたか曖昧	開始点が明確
どこまでやればいいか不明	停止点が事前確定
終わったのか失敗か判別不能	成功判定が自動で決まる

従来の偏差値40層教育	成功体験量産型教育
「高度IT人材育成」で大学に勝とうとする	「挑戦を設計できる人材」を量産
プログラミング能力で評価	設計（Go/Stop）の言語化で評価
成果物の出来を評価	「止めた人」を評価
大きな挑戦をさせる	極端に小さい挑戦を積む

【展開】

Phase 1：成功体験の再定義

> 成功体験は単に「うまくいった記憶」ではありません。本質は、どこから始めて、どこまでやれば、何をもって終わりと言えるかが事前に確定していた経験です。

成功体験が少ない人の実態：

- どこから始めたのか曖昧
- どこまでやればよかったのか分からない
- 終わったのか失敗なのか判別できない

成功体験が少ない人は、挑戦が怖いのではない。 どこから始め、どこで止めれば成功になるのかを、まだ知らないだけだ。

Phase 2: 小さな成功体験の量産設計（10演習）

偏差値40くらいの専門学校生向け、日常生活での小さな成功体験演習：

① 朝の成功を1つ作る（生活）

- Go条件：起きてから10分以内
- Stop条件：1か所（A4一面）が片づいたら終了
- 成功判定：Before/Afterを写真で確認

② あいさつ成功演習（人間関係）

- Go条件：校内で最初に会った人
- Stop条件：返事があってもなくても1人で終了
- 成功判定：声を出した時点で成功

③ 5分予習チャレンジ（学習）

- Go条件：タイマー5分セット
- Stop条件：タイマーが鳴ったら即終了
- 成功判定：1ページでも読めたら成功

④ 分からないを言語化する（学習）

- Go条件：授業後24時間以内
- Stop条件：1文書いたら終了
- 成功判定：「分からない」が言葉になったら成功

⑤ LINE即レス制限（人間関係）

- Go条件：通知が来たら読む
- Stop条件：1通返信したら終了
- 成功判定：それ以上続けない

⑥ 資格勉強1問だけ（学習）

- Go条件：問題集を開く
- Stop条件：1問解いたら終了
- 成功判定：正誤に関係なく1問やったら成功

⑦ 先生に1回だけ質問（人間関係＋学習）

- Go条件：質問文を紙に1文書けたら
- Stop条件：1回聞いたら終了
- 成功判定：答えの内容に関係なく成功

⑧ 今日の「止めてよかった」記録（内省）

- Go条件：就寝前
- Stop条件：1行書いたら終了
- 成功判定：「止めた理由」が書けたら成功

⑨ 予定を途中で切る練習（生活）

- Go条件：タイマー15分
- Stop条件：鳴ったら即停止
- 成功判定：続きを我慢できたら成功

⑩ 週1ミニ挑戦レビュー（統合）

- Go条件：週末に5分確保
- Stop条件：1事例書いたら終了
- 成功判定：開始点と停止点が書けたら成功

Phase 3: レベル2への移行（他者・時間・複合条件）

レベル1：自分一人・短時間・単発で成功を量産 レベル2：他人が絡む、数日〜1週間続く、途中で条件変更が起きる

「最初に決めた条件を、途中で更新できるのが上級者」

レベル2を終えた人の変化：

- 他人が絡んでも疲れにくくなる
- 「続けられない自分」を責めなくなる
- 途中で止めても、次に進める
- 小さな挑戦を設計する癖がつく

Phase 4: 評価設計の転換

この段階で一番やってはいけないのは、「結果が出た／出なかった」で評価すること。ここでそれをやると、全員が一気に黙ります。

重要な運用ルール：

- ❌「頑張ったか」は評価しない
- ❌「結果が良かったか」も評価しない
- ✅ 開始点と停止点を守れたかだけを見る

成功体験とは、うまくいった記憶ではない。 自分で始めて、自分で止められた記憶である。

Phase 5: 偏差値40層のIT専門学校生き残り戦略

偏差値40前後のIT専門学校が生き残る唯一の現実解は、「高尚なIT教育」をやめ、"挑戦を設計できる人材"を量産する学校へ役割転換することです。

現実認識：

- 偏差値40層は成功体験が少ない、自己効力感が低い、抽象論・座学・理想論が通じない
- 「高度IT人材育成」で大学や有名専門に勝つのは不可能

教育設計の3原則：

1. **成功を保証する** – 1回の演習は必ず成功で終わる
2. **止めた人を評価する** – 途中撤退＝高評価
3. **設計だけを評価する** – 開始点／停止点を言語化できたかのみ見る

これにより、臆病な学生は"合理的な挑戦者"に変わる

【結論】

1. **成功体験の再定義** – 「うまくいった記憶」ではなく「開始点と停止点が確定した達成の記憶」
2. **量産設計** – 極端に小さく、必ず成功で終わる演習を10個積み上げる
3. **評価設計の転換** – 結果ではなく設計（Go/Stop）の言語化を評価
4. **グラウンディング優先** – 抽象論の前に、体験で「止められる自分」を作る
5. **偏差値と相関しない能力** – 挑戦を設計できる力は、偏差値や過去の成功体験の差が意味を失う段階に導く

【原文】

→ cases/GoStop設計人材とシリコンバレー東京バンガロール三極連携 – テスト可能な日本語仕様と評価設計の原典.md

---

【メタデータ】

- importance: A
- referenced_count: 0
- last_referenced: null
- source_type: chatgpt_chat
- source_url: https://chatgpt.com/c/69514e30-e458-8322-9d05-8607e5cafc7f
- created_at: 2026-02-23
- conversation_date: 2025-12-24
- related_lenses:
    - GoStop設計人材（Go/Stop条件テンプレートの原典として）

【作成時メモ】

この判例要旨は「グラウンディングと教育原理」レンズから本対話を整理したもの。

骨格文書v3の第一部「グラウンディング（最善）→おまじない（次善）→理論先行（最悪）」の具体的実装として位置づけられる。

「集中講座リファクタリング – 6つの設計原則の発見」が設計原則の体系化であるのに対し、本判例は偏差値40層への具体的な成功体験量産演習を提示している点で、より実践的な補完関係にある。

GoStop設計人材レンズでも別途要旨を作成済み。

授業の「難しさ」分析フレームワーク v2

授業の「難しさ」分析フレームワーク v2

概要

専門学校（ITパスポート等）の授業を「学生目線の難しさ」から評価するためのフレームワーク。 対象学生像：中学校の学力定着が不十分、高校の成績が良くなく専門学校に進学。コンピュータには興味がある。

全体構成

カテゴリ	内容
A. 概念と論理の難しさ	文の構造、前提知識、論理展開など認知プロセスの難しさ
B. 語彙と表現の難しさ	未知語、意味拡張語、表現の揺れなど言葉の難しさ
C. 伝達の物理的問題	声や板書など、伝え方の物理面
D. 定着設計	学んだことを定着させる仕組み
E. コミュニケーション	学生との関係づくり・場の空気
F. メタ学習の指導	「学び方」を教えること

※ カテゴリAは学習者の認知プロセス順に配列している

A. 概念と論理の難しさ

A-1. 文構造（複文・重文・否定・指示語）

一文が複雑だと、文を追うだけで認知資源を使い切る。**最も基礎の土台**。

【複文の問題】
「クライアントがサーバにリクエストを送信すると、
サーバはそれを処理してレスポンスを返す」
├→ 問題：一文に動作が3つ（送信→処理→返す）
├→ 問題：「それ」が何を指すか一瞬迷う
└→ 分解案：
「クライアントがサーバにリクエストを送る。
サーバはリクエストを処理する。
処理が終わったら結果を返す。」
【二重否定の問題】
「パスワードを設定しないわけにはいかない」
├→ 問題：「しない」＋「いかない」で頭が混乱
└→ 言い換え：「パスワードは必ず設定する」
【部分否定の問題】
「すべてのウイルスがファイルを破壊するわけではない」
├→ 問題：「すべて」と「ない」の関係がわかりにくい
├→ 誤解パターン：「ウイルスはファイルを破壊しない」と思う
└→ 言い換え：「ファイルを破壊しないウイルスもある」
【指示語の問題】
「データベースにデータを格納する。それを検索するにはSQLを使う。
これにより効率的な処理が可能になる。」
├→ 問題：「それ」＝データ？データベース？
├→ 問題：「これ」＝SQL？検索？全体？
└→ 言い換え：
「データベースにデータを格納する。
格納したデータを検索するにはSQLを使う。
SQLを使うと効率的に処理できる。」

診断ポイント：一文に動作がいくつあるか？指示語の指示対象は明確か？

教員フィードバック：出題中に二重否定等で誤答があれば、その文法関連の問題を多数演習させたい。

A-2. 概念ツリー（前提知識の要求）

ある概念を理解するために必要な前提概念の連鎖。どこかが欠けていると理解できない。

【例：2進数の計算】
2進数の計算
← べき乗計算
← 割り算と余り
← 掛け算
← 足し算引き算

【例：データベースの正規化】
正規化
← 関数従属性
← 主キーの概念
← テーブルの概念
← 行と列の概念

診断ポイント：この説明は、どの学年の概念を前提としているか？

A-3. 新出概念の密度

一度に多くの新概念が出ると、処理しきれない。

【過密の例】
「TCP/IPでは、データをパケットに分割し、
IPアドレスで宛先を指定し、
ポート番号でアプリケーションを識別し、
ルータがルーティングテーブルを参照して転送する」
├→ 新出概念：パケット、IPアドレス、ポート番号、ルータ、ルーティングテーブル
├→ 問題：1文に5個 → 処理不能
└→ 対策案：
1コマで1〜2概念に絞る
「今日はパケットだけ覚えて帰ろう」
【雪だるま式の崩壊】
1分目：「パケット」が出る → ？
2分目：「パケット」を前提に「ルーティング」 → ？？
3分目：「ルーティング」を前提に「ホップ数」 → ？？？
5分目：完全に迷子
├→ 問題：最初の「？」が解消されないまま積み上がる
└→ 対策案：
1概念ごとに理解確認
「パケットって何だった？」と聞いてから次へ
【適切な密度の目安】
├→ 10分に新出概念1〜2個
├→ 出したら必ず例を出す
├→ 出したら必ず確認する
└→ 次の概念は前の概念が定着してから

診断ポイント：10分間に新出概念はいくつ出ているか？

A-4. 論理展開（飛躍・省略）

教師の頭の中では繋がっていても、学生には見えない論理がある。

【因果の省略】
「HTTPSは安全です。だからネットバンキングに使います」
├→ 省略されている：なぜ安全か
└→ 補完案：
「HTTPSは通信を暗号化する。
暗号化すると他人に読まれない。
だから安全。
だからネットバンキングに使う。」
【前提の省略】
「IPアドレスが枯渇したのでIPv6が作られた」
├→ 省略されている：なぜ枯渇するのか
├→ 省略されている：IPv4の上限は何個か
└→ 補完案：
「IPv4のアドレスは約43億個が上限。
インターネット機器が増えすぎた。
43億では足りなくなった。
だからIPv6が作られた。」
【飛躍】
「OSがないとアプリは動きません」
├→ 飛躍：なぜ動かないのか
├→ 学生の頭：「動かないって言われても…」
└→ 補完案：
「アプリは画面表示やファイル保存をOSに頼んでいる。
OSがいないと頼む相手がいない。
だから動けない。」

診断ポイント：AだからBと言っているが、間に省略された論理はないか？

A-5. 具体と抽象の接続

抽象的な説明だけでは「だから何？」となる。具体化して腑に落ちる段階。

【抽象だけの例】
「OSI参照モデルは7層で構成され、
各層が独立した役割を持ち、
上位層は下位層のサービスを利用する」
├→ 問題：全部抽象。何も見えない。
├→ 学生の頭：「だから何？」「で？」
└→ 対策：具体例から入る
【具体から入る例】
「YouTubeを見るとき何が起きてるか？
①君がクリックする
②パソコンが『動画くれ』と送る

③サーバが動画を送り返す
④パソコンが動画を画面に出す

これを整理したのがOSI参照モデル」
├→ 具体（YouTube）→ 抽象（OSI）の順
└→ 「ああ、あれを整理したものか」となる
【例が遠い問題】
「プロトコルは外交儀礼のようなものです」
├→ 問題：外交儀礼を知らない
├→ 例が学生の生活世界にない
└→ 近い例：
「LINEで最初に『おつ』って送るやん？
相手も『おつ』って返すやん？
それがプロトコル。最初の挨拶の決まり事。」

診断ポイント：抽象的説明に具体例があるか？その例は学生の生活世界にあるか？

B. 語彙と表現の難しさ

B-1. 未知語（学年配当外の語彙）

小中学校で習わない言葉が説明なしに出てくると、そこで止まる。

【脆弱性】
├→ セキュリティホール
│　　→ 専門用語→専門用語（✗ 余計難しい）
├→ 突かれると乗っ取られる
│　　→ 「つかれる」が「疲れる」に聞こえる（✗ 誤解リスク）
└→ 悪いやつに乗っ取られる弱み／プログラムが暴走する弱点
　　→ 日常語＋具体的結果（○ 理解可能）
【暗号化】
├→ エンクリプション
│　　→ 英語（✗ さらに遠い）
├→ 鍵をかける
│　　→ 比喩として近い（△ 入口としてはOK）
└→ 他人に読めないようにぐちゃぐちゃにする
　　→ 日常語＋結果がわかる（○）
【プロトコル】
├→ 通信規約
│　　→ 漢語（✗「規約」も難しい）

├→ ルール
│　　→ 抽象的すぎる（△ 何のルール？）
└→ コンピュータ同士が話すときの「お作法」「決まり事」
　　→ 日常語＋比喩（○）

診断ポイント：この単語は小中学校で習うか？習わないなら授業内で説明があるか？

B-2. 偽の既知語（文脈による意味拡張）

知っている言葉だと思っているが、IT文脈では意味が違う／広い。学生は「わかったつもり」でスルーするため、未知語より厄介。

【実行】
├→ 文系文脈：作業を行う、計画を実行に移す
└→ IT文脈：プログラムを動かす、システムを走らせる
　　→ 「実行ボタン」「実行ファイル」
【処理】
├→ 文系文脈：事務処理、手続きを済ませる
└→ IT文脈：データに対して計算・変換を行う
　　→ 「処理速度」「バッチ処理」
【環境】
├→ 文系文脈：自然環境、生活環境
└→ IT文脈：ソフトが動く条件・設定の総体
　　→ 「開発環境」「実行環境」「環境変数」
【資源】
├→ 文系文脈：石油、水、天然資源
└→ IT文脈：CPUやメモリなど、システムが使うもの
　　→ 「リソース不足」「資源の解放」
【手続き】
├→ 文系文脈：役所での手続き、申請手続き
└→ IT文脈：処理の手順、プロシージャ
　　→ 「手続き型言語」
【入力／出力】
├→ 文系文脈：データを入れる／出す（漠然）
└→ IT文脈：厳密に定義されたI/O操作
　　→ 「入力デバイス」「標準出力」
【変換】
├→ 文系文脈：形を変える（漠然）
└→ IT文脈：データ形式を別の形式に変える
　　→ 「型変換」「文字コード変換」
【階層】
├→ 文系文脈：組織の上下関係
└→ IT文脈：データやシステムの構造的な段階

→ 「ディレクトリ階層」「OSI参照モデルの階層」
【冗長】
├→ 文系文脈：無駄が多い（ネガティブ）
└→ IT文脈：予備を持たせて安全性を高める（ポジティブ）
→ 「冗長化」「冗長構成」
※ 価値の逆転があるため特に混乱しやすい
【透過】
├→ 文系文脈：光が透き通る
└→ IT文脈：ユーザーから見えない・意識させない
→ 「位置透過性」「透過的アクセス」

診断ポイント：この言葉は日常語と同じ意味か？IT文脈での意味拡張を明示しているか？

B-3. 表現の揺れ（同一概念の異表現）

同一概念を指すのに表現がバラバラだと、学生は別概念だと思う。

【同一概念の表現ブレ】
「分解」「分ける」「ばらばらにする」「切り分ける」
├→ 教師の頭：全部同じ意味
├→ 学生の頭：4つの別概念に見える
└→ 対策：用語を統一するか、「同じ意味だよ」と明示
【専門用語と日常語の混在】
「保存」「セーブ」「格納」「ストア」
├→ 1回目：「ファイルを保存して」
├→ 2回目：「データをストアする」
├→ 3回目：「DBに格納」
├→ 学生の頭：3つの違う操作？
└→ 対策：「保存、セーブ、格納、ストア、全部同じ」と最初に宣言
【略語と正式名称の混在】
「OS」「オペレーティングシステム」「基本ソフト」
├→ 問題：3回出てきて3回とも別物に見える
└→ 対策：「OS、正式にはオペレーティングシステム、
日本語では基本ソフト。今日からOSで統一」
【動詞の揺れ】
「実行する」「走らせる」「動かす」「起動する」
├→ 微妙に意味が違う場合もある
├→ 同じ意味で使っている場合もある
├→ 学生：区別すべき？しなくていい？
└→ 対策：違いがあるなら明示、ないなら統一

診断ポイント：同じ概念を指す言葉が複数出ていないか？出ているなら同一であることを明示しているか？

教員フィードバック：同じ概念のものを選ぶ／違う概念のものを選ぶ問題で、同一概念グループの語彙を整理させたい。

C. 伝達の物理的問題

C-1. 早口

├→ 1分間に400字以上 → 聞き取れない
├→ 専門用語を早口で言う → 音として認識できない
│　　例：「ディーエイチシーピー」→「でぃーえっちぴー？」
└→ 対策：
　　・専門用語は特にゆっくり
　　・重要な点は2回言う
　　・間を取る

C-2. 声が単調

├→ 同じトーン、同じ速度で15分 → 眠くなる
├→ 重要な点も雑談も同じ声 → メリハリがない
└→ 対策：
　　・重要な点で声を大きく／ゆっくり
　　・「ここ大事」と明示
　　・質問を挟んで空気を変える

C-3. 板書をすぐ消す

├→ 学生がノートに写す時間がない
├→ 振り返ろうとしたら消えている
└→ 対策：
　　・消す前に「写した？」と確認
　　・スライドなら後で配布

C-4. 板書の字が汚い／小さい

├→ 読めない → 写せない → 残らない
├→ 略字や癖字 → 誤読
│　　例：「ア」と「マ」、「ソ」と「リ」
└→ 対策：
　　・大きく書く
　　・楷書で書く
　　・重要語は色を変える

C-5. 図と説明の不一致

├→ 図を指さしながら説明するが、どこを指しているか見えない
├→ 「ここ」「これ」が図のどこかわからない
└→ 対策：
・「左上の四角」など言葉で位置を示す
・レーザーポインタの動きを遅く

診断ポイント：声の速さ・抑揚は適切か？板書は読めるか？図と説明は対応しているか？

D. 定着設計

D-1. 演習問題の量が少ない

├→ 説明30分 → 演習5分 → 次の単元
├→ 「わかった気がする」で終わる
├→ 実際に手を動かしていない → 定着しない
└→ 対策：
・説明15分 → 演習15分 のバランス
・「やってみて」の時間を確保

D-2. 復習がない

├→ 1回やって終わり
├→ 1週間後：「やったっけ？」
├→ 忘却曲線：1日後には74%忘れる
└→ 対策：
・翌週の冒頭で前回の振り返り5分
・「先週やったXXX、覚えてる？」から始める

教員フィードバック：Google Formもよいが、Moodleの問題バンク機能を使ったほうが効果的ではないか。

D-3. 重要概念が絞られていない

├→ 「全部重要」＝「何が重要かわからない」
├→ 20個の用語が同じ重みで出てくる
├→ 結果：全部曖昧に覚えて全部曖昧に忘れる
└→ 対策：
・「今日は3つだけ覚えて」と絞る
・「これは参考程度」「これは必須」を明示

D-4. 反復利用がない（一度出て終わり）

├→ 第3回で「IPアドレス」を教える

├→ 第4回以降で出てこない
├→ 結果：孤立した知識 → 忘れる
└→ 対策：
・後の単元で意図的に既出概念を使う
・「第3回でやったIPアドレス、ここで使うよ」
・概念のネットワークを作る

診断ポイント：演習の時間は十分か？復習の機会はあるか？重要概念は絞られているか？

E. コミュニケーション

E-1. 緊張をとく（笑い）

├→ 効果：場が和む → 質問しやすくなる
├→ 失敗パターン：
│　・滑る → 逆に気まずい
│　・内輪ネタ → わからない人が疎外感
└→ コツ：
・自虐は安全（「先生も最初わからんかった」）
・学生あるあるネタ（「Wifiないと死ぬよな」）

E-2. 同じ目線に立つ

├→ 効果：「この先生はわかってくれる」→ 安心
├→ 失敗パターン：
│　・上から目線：「こんなの簡単だろ」
│　・馬鹿にした態度：「まだわからないの？」
└→ コツ：
・「これ難しいよな」と共感
・「先生も昔ここで躓いた」と体験を共有

E-3. 褒めて乗せる

├→ 効果：自己効力感 → もっとやろうとする
├→ 失敗パターン：
│　・嘘っぽい褒め：「すごーい」（棒読み）
│　・的外れな褒め：努力してないのに「頑張ったね」
└→ コツ：
・具体的に褒める：「この発想はよかった」
・プロセスを褒める：「ちゃんと手順踏んだな」

E-4. 怒って引き締める

├→ 効果：緊張感 → 集中
├→ 失敗パターン：
│　　・感情的に怒る → 恐怖 → 萎縮
│　　・人格否定：「お前は駄目だ」
│　　・長い説教 → 響かない
└→ コツ：
　　・行動を指摘：「今の態度はまずい」
　　・短く、切り替える
　　・怒った後のフォロー

診断ポイント：場の空気を読んでいるか？学生との関係づくりができているか？

F. メタ学習の指導

F-1. 目標を持たせる

├→ 効果：「何のためにやってるか」がわかる → 動機
├→ 失敗パターン：
│　　・「とりあえず覚えろ」→ 意味がわからない
│　　・目標が遠すぎる：「将来役立つ」→ 実感ない
└→ コツ：
　　・近い目標：「来週の小テストで7割」
　　・具体的なご褒美：「これができたら次の単元が楽になる」

F-2. 復習方法を教える

├→ 効果：自分で定着させられる → 自立
├→ 問題：「復習しろ」だけでは何をすればいいかわからない
└→ 具体的指示の例：
　　・「寝る前に今日のノートを1分見返せ」
　　・「土曜に演習問題をもう1回解け」
　　・「用語カードを作って電車で見ろ」

F-3. 興味の深堀りを促す

├→ 効果：主体的学習 → 知識が広がる
├→ 失敗パターン：
│　　・「自分で調べろ」だけ → 何を調べるかわからない
└→ コツ：
　　・入口を示す：「YouTubeで○○で検索すると面白い動画ある」
　　・問いを投げる：「なんでこうなってると思う？調べてみ」

F-4. 自力努力の仕方を教える

├→ 問題：「頑張れ」では頑張り方がわからない
├→ 学力が低い学生は「勉強の仕方」を知らないことが多い
└→ 具体的指示の例：
- 「まず例題を写す。次に数字を変えて解く」
- 「わからなかったら3分考えて、だめなら答え見ていい」
- 「1回で覚えようとするな。5回やれ」

診断ポイント：学習の目標は明示されているか？学び方そのものを教えているか？

分析可能性マトリクス

領域	テキスト分析	音声分析	映像分析	学生データ
A. 概念と論理	◎			
B. 語彙と表現	◎			
C. 伝達の物理	△	○	○	
D. 定着設計	○			◎
E. コミュニケーション	△	○	◎	
F. メタ学習指導	○			◎

開発の優先順位

【第一段階：テキストベースで確実にできること】
→ A（概念と論理）、B（語彙と表現）
【第二段階：音声を加える】
→ C の一部（早口、単調さ）
【第三段階：カリキュラム分析】
→ D（定着設計）
【第四段階：高度な分析】
→ E, F（コミュニケーション、メタ学習）

→ 映像分析 or 学生フィードバックが必要

復習の設計（教員フィードバック）

文構造（A-1）に対応する演習

【目的】
二重否定・部分否定・指示語・複文・重文の理解を定着させる
【方法】
・出題の中に二重否定等が出てきて誤答があれば、その文法項目の問題を集中的に演習
・同じ文法パターンの問題を多数用意して反復練習
【問題例：二重否定】
「次の文を肯定文に言い換えなさい」
・パスワードを設定しないわけにはいかない → パスワードは必ず設定する
・セキュリティ対策をしないことはない → セキュリティ対策をする
【問題例：部分否定】
「次の文の意味として正しいものを選びなさい」
・すべてのウイルスがファイルを破壊するわけではない
A: ウイルスはファイルを破壊しない
B: ファイルを破壊しないウイルスもある ←正解
【問題例：指示語】
「『それ』が指すものを答えなさい」
・データベースにデータを格納する。それを検索するには… → データ

語彙・表現（B-3）に対応する演習

【目的】
同一概念グループの語彙を整理し、表現の揺れに惑わされないようにする
【方法】
・同じ概念のものを選ぶ問題
・違う概念のものを選ぶ問題
・グルーピング問題
【問題例：同じ概念を選ぶ】
「次の中から『データを保存する』と同じ意味の表現をすべて選びなさい」
☐ データをセーブする ←正解
☐ データを格納する ←正解
☐ データをストアする ←正解
☐ データを転送する
【問題例：仲間外れを選ぶ】
「次の中から意味が異なるものを1つ選びなさい」
A: 実行する
B: 走らせる

C: 起動する
D: 終了する ←正解
【問題例：グルーピング】
「次の語を同じ意味のグループに分けなさい」
保存、削除、セーブ、消去、格納、除去
→ グループ1（保存系）：保存、セーブ、格納
→ グループ2（削除系）：削除、消去、除去

復習システムの選択

【Google Form】
・手軽に作成できる
・集計が自動
・URLを共有するだけで使える
【Moodle問題バンク】（教員推奨）
・問題をカテゴリ別に蓄積できる
・ランダム出題が可能
・学生ごとの正答率を追跡できる
・誤答パターンに応じた追加問題を出せる
・スペースド・リピティション（間隔反復）に対応しやすい
【運用案】
1. 問題バンクに文法項目別・語彙グループ別に問題を蓄積
2. 小テストで誤答があった項目を特定
3. その項目の追加問題を自動または手動で割り当て
4. 定着するまで反復

今後の課題

1. **概念ツリーの構築**：学習指導要領ベースで小中高の概念階層を整理
2. **語彙辞書の構築**：
    - 教育漢字＋学年配当語彙のデータベース化
    - 偽の既知語（文系→IT意味拡張語）のリスト化
3. **文構造分析プロンプト**：過去に作成したプロンプトの再利用・改良
4. **ITパスポート教科書の分析**：各単元の難しさマップ作成
5. **演習問題システム**：
    - A-1（文構造）：二重否定・部分否定・指示語の演習問題
    - B-3（表現の揺れ）：同一概念グループの語彙整理問題
    - Moodleの問題バンク機能の活用検討

変更履歴

v1 → v2

- **A（概念と論理の難しさ）を再編**
  - 旧A-2（語彙）をBに移動
  - 配列を学習者の認知プロセス順に変更
- **B（語彙と表現の難しさ）を拡張**
  - B-1：未知語（旧A-2の一部）
  - B-2：偽の既知語（新規追加、教員フィードバック反映）
  - B-3：表現の揺れ（旧B）
- **教員フィードバックを反映**
  - A-1, B-3, D-2に演習・復習に関するコメント追加

コネクト法

コネクト法

「わかる → 覚える → 考える」授業設計枠組み

清風情報工科学院

SEIFU Institute of Information Technology

2026年

1. コネクト法とは何か

1-1. 核心的な問い

「覚えられない」と悩む学生は多い。しかし問題の本質は記憶力ではなく、「覚え方の順序」にある。

コネクト法は、この問いから出発している。

> **「覚えられない」の正体は、記憶力の問題ではない。**
> 「わかる」をすっ飛ばして「覚える」をやっているから入らへんねん。

1-2. 「わかる」の定義

「わかる」とは、わかっていないことを、わかっていることに接続することである。

つまり、未知の情報を既知の情報に繋げた瞬間に「わかった」が起きる。逆に言えば、接続先がなければ何度繰り返しても「わかってないまま」になる。

状態	結果
接続先がある場合	「あ、あれと同じやん」という感覚が生まれる → 覚えられる
接続先がない場合	呪文にしか聞こえない → 何度繰り返しても入らない

1-3. 学習の3ステップ

コネクト法では、学習を以下の3ステップで捉える。

① コネ探し　—　接続先を見つける

わかっていることと、新しい概念を繋げる接続点を探す作業。
「これは何かに似ていないか？」「日常でいうとどんな場面か？」という問いが鍵。
ここが成立して初めて、次のステップが意味を持つ。

② コネくり　—　こねくり回して定着させる

接続が成立した情報を、反復・アウトプット・言い換えによって定着させる。
間隔をあけた復習（翌日・1週間後・2週間後）が効果的。

「書く・読む」のインプットだけでなく、「説明する・問題を解く」アウトプットが不可欠。

③ コネ使い　—　繋がった網で考える

複数の接続が網になり、「考える」が可能になった状態。
これが最終目標。暗記の先にある「応用・問題解決・創造」の入口。
「覚えること」から「使うこと」へのシフトが起きる段階。

2. 教員向け授業設計ガイドライン

2-1. 授業設計の基本原則

すべての授業において、「コネ探し → コネくり → コネ使い」の順序を意識した設計を行う。特に「コネ探し」の設計が最重要であり、ここを教員任せ・偶然任せにしないことが品質の鍵となる。

2-2. 授業計画書への記載事項

各授業の計画書に以下の3項目を明記することを標準とする。

項目	内容
① 新概念の特定	今回の授業で学生にとって「未知」となる概念・用語を列挙する
② 接続先の設計	各概念を「何と繋げるか」を具体的に記述する（日常体験・既習知識・比喩など）
③ 確認方法	接続が成立したかどうかをどう確認するか（発問・ミニテスト・言い換えなど）

2-3. 接続先の設計パターン

接続先には以下のパターンがある。授業の内容に応じて最適なものを選ぶ。

パターン	例
日常体験への接続	「ネットワークのルーターは、郵便局と同じ仕組みやで」
既習知識への接続	「これ、先週やったあの概念の応用やから」
身体感覚への接続	「ループ処理は、洗濯物を1枚ずつ干す動作と同じ」
ストーリーへの接続	一連の流れをキャラクターや物語として語る
図・絵への接続	抽象概念を視覚的な図に落とし込む

2-4. 「コネ探しをすっ飛ばしている」授業のサイン

以下のような状況が続く場合、コネ探しが機能していない可能性がある。

- 学生が「何を言っているかわからない」という反応を繰り返す
- 同じことを何度教えても定着しない
- テストは解けるが、別の文脈では使えない

- 学生が「覚えられない、自分は阿呆や」と自己否定している

こうした場合は「反復を増やす」のではなく、「接続先の設計から見直す」ことが先決である。

3. 学生向け「学び方」オリエンテーション

3-1. 最初に伝えること

学期の最初に、すべての学生に以下の認識転換を届ける。

> **覚えられへんのは、頭が悪いからちゃう。**
> コネ探しをすっ飛ばして、コネくりだけやってるから入らへんねん。

3-2. 自分のコネ探しの方法を知る

人によって「接続しやすい感覚」は異なる。自分がどのタイプかを把握することで、コネ探しの効率が上がる。

タイプ	特徴
視覚タイプ	図・絵・色・レイアウトで繋げると入りやすい
聴覚タイプ	声に出す・リズムに乗せる・説明を聞くと入りやすい
体験タイプ	実際にやってみる・日常の行動に例えると入りやすい
物語タイプ	ストーリーや文脈の中に置くと入りやすい

どのタイプかは、「何かを覚えているとき、頭の中でどんな映像・音・感覚が動いているか」を観察することでわかってくる。

3-3. 学生自身が使えるコネ探しの問い

新しいことを学ぶとき、まず自分にこう問いかけてみる。

- これは日常でいうと、何に似てるやろ？
- 前に習ったこととどこかつながってへんか？
- これを誰かに説明するとしたら、どんな言葉を使う？
- もし図に描くとしたら、どんな形になる？

この問いに答えられたとき、「コネ探し」が成立している。そこから初めて反復・復習が意味を持つ。

4. AI個別学習ツールの設計思想

4-1. 基本コンセプト

コネクト法に基づくAI学習ツールは、「正解を教えるAI」ではなく「コネを引き出すAI」として設計する。Human in the Loopを前提とし、AIが足場を組みながら学生が自分で接続先を発見するプロセスを支援する。

4-2. 3フェーズの設計

フェーズ	AIの役割
フェーズ1コネ探しのAI支援	AIが「あなたはこれを何かに例えると何ですか？」「日常でいうとどんな場面に似ていますか？」と問いかけ、学生の既知を引き出す。接続の確認ができて初めて次に進む。
フェーズ2コネくりのAI支援	接続が成立した情報に対して、間隔復習・ミニテスト・言い換え練習をAIが設計・実施する。学生の回答を分析し、理解度に応じて問題難易度を調整する。
フェーズ3コネ使いのAI支援	応用問題・シナリオベースの課題を通じて、知識を「使う」体験を提供する。誤答に対しては「どこのコネが切れているか」をフィードバックする。

4-3. Human in the Loopの設計原則

AIは学習の補助者であり、人間（教員・学生）が主体であることを設計の中心に置く。

- 教員は授業設計の段階で「コネ探しの接続先」をAIに登録・更新できる
- AIの問いかけに対する学生の回答は教員に可視化され、指導の参考になる
- AIが「接続に失敗している」と判断した場合は、教員へのアラートを出す
- AIはあくまで足場であり、フェーズ3では学生がAIなしで考えられることを目標とする

4-4. 段階的自立のモデル

AI活用は学生の依存を生むのではなく、自立を促す方向に設計する。

時期	役割の変化
入学初期	教員がコネ探しを設計・提示し、学生は「接続の感覚」を体験で学ぶ
中期	AIが問いかけ、学生が自分でコネを探す。教員はモニタリングと補助

後期	学生が自力でコネ探しをできるようになる。AIは確認ツールとして機能

5. コネクト法　全体図

コネクト法　3ステップ × 3層

① コネ探し　（わかる）

教員が接続先を設計・提示　／　AIが問いかけてコネを引き出す　／　学生が自力でコネを探す

② コネくり　（覚える）

間隔復習　／　アウトプット練習　／　言い換え・説明

③ コネ使い　（考える）

応用・問題解決・創造　／　知識の網を使って考える

コネクト法の根本思想

人はわかるところまで降りたら、必ずわかる。
そこから自分の知識を確実に増やしていける。
「覚えられない」は能力の問題ではなく、順序の問題である。

J検 情報活用試験3級

J検 情報活用試験3級

「満点合格」学習戦略ドキュメント【統合版】

目的：J検3級を「暗記の練習台」として活用し、**満点合格**を目標に「覚え方のコツ」と「読み方のコツ」を体得させる。

この戦略の2つの柱：

- **柱1. 暗記技術**—— 400語の専門用語を体系的に定着させる方法論
- **柱2. 読解技術**—— テキストと問題文の日本語を読み解く力の底上げ

用語を完璧に覚えても、テキストの説明文が読めなければ自習できない。問題文が読めなければ「何を聞かれているか」がわからず正答できない。暗記技術だけでは満点は取れない。読解技術という土台があって初めて暗記が機能する。

構造：読解技術（土台）→ 落とし穴マップ（混乱の除去）→ 暗記技術（定着）→ 満点合格

1. テキスト全体像の数値分析

項目	数値	備考
総用語数（索引ベース）	約400語	英字略語150〜170語＋和文230〜250語
章数	5章	第2章が全体の約4割を占める
章末問題数	16問（小問約80問）	穴埋め6割、正誤3割、マッチング1割
計算問題	実質2パターン	補助単位換算、データ量概算
合格に必要な力	用語⇔意味の双方向対応＋問題文の読解	論述・計算過程記述は一切なし

結論：「400語を正確に覚え、問題文が正確に読めたら満点が取れる」試験。知識の暗記と日本語の読解の両面が必要。

2. 章別の特徴と難易度

章	用語数	暗記の性質	落とし穴の中心	暗記レベル

第1章 パソコンの基礎	約150語	装置名・規格名の対応暗記	A-3 新出概念の密度	Lv2: 対応暗記
第2章 ネットワーク	約130語	理解＋暗記（プロトコル・サーバ）	B-1 略語の洪水	Lv3: 理解＋暗記
第3章 アプリケーション	約40語	ソフトの種類と機能の対応	B-3 表現の揺れ	Lv1: 日常語に近い
第4章 情報社会	約50語	システム名・概念の対応	B-2 偽の既知語	Lv3: 理解＋暗記
第5章 情報モラル	約30語	ツリー構造の暗記（権利体系）	比較的少ない	Lv1: 入門に最適

推奨する学習順序：第5章（入門）→ 第3章（日常語に近い）→ 第1章前半（装置名）→ 第4章（社会システム）→ 第2章（最難関）→ 第1章後半（インタフェース・計算）

教科書の章順ではなく、暗記難易度の低い順に並べ替える。

3. 落とし穴マップ ―― 専門用語の障壁

フレームワークの分析軸に基づき、テキスト本文から専門用語に関する落とし穴を洗い出した。

3-1. B-1 略語の洪水（最大の障壁）

第2章だけで英略語が20個以上出る。学生にはアルファベットの羅列に見えて区別がつかない。

対策：「係の名前」方式 ―― 日本語で先に覚えてから英略語を紐づける

略語	「何をするやつか」（一言）	正式名称
SMTP	メールを送る係	Simple Mail Transfer Protocol
POP	メールを受け取る係	Post Office Protocol
IMAP	メールをサーバで管理する係	Internet Message Access Protocol
DNS	名前→番号の翻訳係	Domain Name System
DHCP	番号を自動で配る係	Dynamic Host Configuration Protocol
FTP	ファイルを運ぶ係	File Transfer Protocol
HTTP	Webページを運ぶ係	HyperText Transfer Protocol
HTTPS	Webページを暗号化して運ぶ係	HTTP Secure
TLS/SSL	通信を暗号化する鍵係	Transport Layer Security
NTP	時計を合わせる係	Network Time Protocol
MIME	メールに画像や動画をつける係	Multipurpose Internet Mail Extensions
TCP	届いたか確認する配達係	Transmission Control Protocol
IP	住所（IPアドレス）を管理する係	Internet Protocol

サーバも同じ方式で整理する：

サーバ名	「何をする場所か」	プロトコルとの関係
WWWサーバ（Webサーバ）	Webページを配る場所	HTTP/HTTPSを使う
SMTPサーバ	メールを送り出す郵便局	SMTPを使う
POPサーバ	届いたメールを保管する場所	POPを使う
DNSサーバ	名前と番号の対応表がある場所	DNSを使う
DHCPサーバ	番号を配る受付	DHCPを使う

FTPサーバ	ファイル置き場	FTPを使う
proxyサーバ	代理で外に出る係	クライアントの代理
ファイアウォール	門番（不正を止める壁）	通信の監視・制御
ゲートウェイ	通訳（異なるルール間の橋渡し）	プロトコル変換
ストリーミングサーバ	動画・音楽を流し続ける放送局	データを少しずつ配信

3-2. B-2 偽の既知語（わかったつもりの罠）

日常語と同じ言葉がIT文脈では意味が変わる。学生は「知ってる言葉」としてスルーするため、未知語より厄介。

用語	日常での意味	IT文脈での意味	テキスト内の出現例
処理	事務を済ませる	データに計算・変換を行う	処理速度、バッチ処理
環境	自然環境、生活環境	ソフトが動く条件・設定の総体	実行環境、開発環境
実行	計画を行う	プログラムを動かす	実行ファイル(.exe)
冗長	ムダが多い（ネガティブ）	予備があって安全（ポジティブ）	RAID＝冗長化
階層	組織の上下関係	データ構造の段階	ディレクトリ階層
資源	石油・水	CPU・メモリなどシステムが使うもの	コンピュータ資源の共有
情報	漠然と「ニュース」	JIS定義：特定の文脈で意味を持つもの	テキストp11の定義
データ	漠然と「数字」	情報を形式化したもの	テキストp11の定義
プロトコル	（知らない言葉）	通信の「お作法」「決まり事」	TCP/IP等すべて
認証	漠然と「確認」	正規の利用者かどうか確かめること	ユーザID＋パスワード

3-3. B-3 表現の揺れ：IT用語（暗記量が膨れる罠）

同じ概念を指す複数の表現が整理されないと、学生は別の概念だと思って暗記量が倍増する。

グループ	同一概念の表現バリエーション	授業での宣言例
OS	オペレーティングシステム / 基本ソフトウェア / 基本ソフト	「全部同じ。授業ではOSで統一」
アプリ	アプリケーションソフトウェア / 応用ソフトウェア / アプリケーションソフト / アプリ	「全部同じ。アプリで統一」

CPU	中央処理装置 / Central Processing Unit / 制御装置＋演算装置	「CPUは制御と演算の合体」
メインメモリ	主記憶装置 / メインメモリ	「同じもの」
HDD	ハードディスク / HD / HDD / Hard Disk Drive	「全部同じ。HDDで統一」
LAN	ローカルエリアネットワーク / Local Area Network	「LANで統一」
ISP	インターネットサービスプロバイダ / プロバイダ / ISP	「プロバイダで統一」
GUI/CUI	グラフィカルユーザインタフェース / キャラクタユーザインタフェース	「画面操作=GUI、文字操作=CUI」

3-4. 紛らわしいペア（混同の罠）

試験で間違えやすい「似ているが違うもの」を比較表で対にして覚える。単独で覚えると必ず混同する。

ペア	Aの特徴	Bの特徴	覚え方のヒント
LAN vs WAN	建物内の近いネットワーク	遠隔地をつなぐネットワーク	L=Local / W=Wide
POP vs IMAP	メール全部受信、端末に保存	メールはサーバで管理	POP=持ってくる / IMAP=置いとく
NAT vs NAPT	IPアドレスを1対1変換	IPアドレスを1対多変換	Pが付く=Port（多い）
RGB vs CMY	光の三原色（画面）	色の三原色（印刷）	RGB=混ぜると白 / CMY=混ぜると黒
加法混色 vs 減法混色	光を足すと白に近づく	インクを足すと黒に近づく	加法=光 / 減法=インク
コンパイラ vs インタプリタ	一括翻訳→実行（速い）	1行ずつ翻訳→実行（手軽）	コンパイラ=まとめて / インタプリタ=ちょっとずつ
絶対パス vs 相対パス	/から始まる（ルートから）	今いる場所から辿る	絶対=/スタート / 相対=ここから
BtoB / BtoC / CtoC	企業↔企業	企業→消費者 / 消費者↔消費者	B=Business / C=Consumer
ディレクトリ型 vs ロボット型	人手で情報収集・分類	プログラムで自動収集	ディレクトリ=人 / ロボット=機械
オプトイン vs オプトアウト	許可してから送る	拒否されるまで送る	イン=許可で入る / アウト=断って出る
シリアル vs パラレル	1ビットずつ送る	複数ビットまとめて送る	シリアル=1列 / パラレル=並列

4. 落とし穴マップ ―― 日本語の壁（読解力の障壁）

専門用語を全部覚えても、テキストの日本語が読めなければ自習できず、問題文が読めなければ正答できない。ここでは「専門用語」ではなく「日本語そのもの」に起因する障壁を整理する。

4-1. A-1 文構造の複雑さ ―― テキストに出る「読めない文」の4パターン

テキストの説明文と章末問題の選択肢には、一文50字超の複文・重文が頻出する。学力の弱い学生は「文の途中で迷子になる」。

パターン①：定義文の入れ子構造

テキスト第1章 p11「情報」のJIS定義：

「事実，事象，事物，過程，着想などの対象物に関して知り得たことであって，概念を含み，一定の文脈中で特定の意味を持つもの」

問題：連体修飾が3層。並列か入れ子か判断できない

分解法：読点で区切る → 骨格（主語＋述語）だけ抜く → 一言にする

→ **「ある場面で意味がある知識」＝ 情報**

パターン②：条件＋結果の複文（問題文に多い）

章末問題 第4章 問1(4)：

「ETCは，自動車が料金所を通過するときに，ゲートとカーナビの間で交信して自動的に料金の支払い手続きを行う」

問題：定義・条件・動作が一文に圧縮。正誤判定ポイント（カーナビ→正しくはETC車載器）が文の中盤に埋没

分解法：「誰が → 何を → どうする」を抜く → 残りの修飾語をチェック

パターン③：否定＋例外の文（正誤問題の罠）

典型的な正誤問題：

「著作権は，著作物を創作した時点で自動的に発生し，特許権のように出願・登録の手続きを必要としない」

問題：肯定（発生し）と否定（必要としない）が混在し、比較（特許権のように）が挿入される

分解法：肯定部分と否定部分を分離する → 「作った瞬間に権利発生。届出不要」

パターン④：列挙＋修飾語（テキスト本文に多い）

テキスト第1章 入力装置の説明：11種類が一文で列挙される

問題：途中で「今どこを読んでいるか」を見失う。末尾のOCRとOMRを混同する

分解法：指で数えながら読む → グループに分ける（手で操作 / 画像取込 / 印刷物読取）

毎週の授業冒頭10分で1文だけ分解する練習を行う。15週で15文の実践訓練。文法用語（主語・述語・連体修飾語…）は使わない。「誰が？何を？どうする？」の3つだけ。

4-2. B-3拡張：日本語レベルの表現揺れ

IT用語の揺れ（3-3で整理済み）に加え、テキストの「地の文」でも同じ概念に複数の動詞が使われる。整理しないと、学生は読むたびに「新しいこと」だと思い、認知負荷が膨れ上がる。

同義グループ	テキストで使われるバリエーション	統一する表現
保存する系	保存する / セーブする / 格納する / 記憶する / 蓄積する / 記録する / 保管する	「保存する」
使う系	利用する / 使用する / 用いる / 活用する / 適用する	「使う」
送る系	送信する / 送る / 転送する / 伝送する / 配信する / 発信する	「送る」（転送＝別の人に回す、配信＝多くの人に配る、は区別）
変換する系	変換する / 変える / 置き換える / 翻訳する / コンバートする	「変換する」
つなぐ系	接続する / 繋ぐ / リンクする / 連携する / アクセスする	「つなぐ」（アクセス＝通信して使う、は別概念）
守る系	保護する / 防ぐ / 防止する / 防御する / セキュリティを確保する	「守る」
管理する系	管理する / 運用する / 制御する / コントロールする / 運営する	「管理する」（制御＝機械的操作、は区別）
処理する系	処理する / 実行する / 演算する / 計算する	「処理する」（演算＝計算、と明示）

4-3. 定義文のパターンの揺れ

テキストは新しい概念を定義するとき様々な言い回しを使う。学生は「〜という」は定義と認識できるが、「〜と呼ばれている」では気づかない。

定義パターン	テキスト内の例	意味
Aは Bという	「サービス事業者をISPという」	A＝B（最も基本）
Aは Bと呼ぶ/呼ばれる	「この仕組みをクライアントサーバシステムと呼ぶ」	A＝B
Aは Bともいう	「基本ソフトウェアともいう」	Aの別名がB
Aは Bのことである	「ドメイン名とはIPアドレスに対応する名前のことである」	A＝B

A すなわち B	「TCP/IP，すなわちインターネットの標準プロトコル」	言い換え
A（B）	「WWW（World Wide Web）」	カッコ内は正式名称/略称
Aは，Bする（もの/仕組み/方式）	「ファイアウォールは，不正アクセスを防止するもの」	Aの機能がB

授業での宣言：「太字の前後に必ず上のどれかの言い回しがある。太字を見たら『＝の表現』を探せ」

5. 問題文の読解技術

章末問題の出題形式は3パターンしかない。形式ごとに「何を聞かれているか」「どこを見ればいいか」を明示的に教える。

5-1. 問題文の「命令語」辞典

問題文そのものに使われる日本語がわからないと、何をすべきか理解できない。以下を「試験の用語」として事前に教える。

問題文に出る表現	意味（やさしく言い換え）	具体例
「適切なものを選べ」	正しいものを1つ選びなさい	消去法が使える
「適切でないものを選べ」	間違っているものを1つ選びなさい	3つは正しい。残り1つが答え
「記述中の〔　〕に入れるべき適切な字句を…」	文の空欄に当てはまる言葉を選べ	「字句」＝言葉のこと
「解答群から選べ」	下の選択肢リストから選びなさい	「解答群」＝選択肢の一覧
「…に関する記述として」	～についての説明として	「に関する」＝「について」
「以下の〈条件〉のもとで」	この前提で考えなさい	条件を先に読むこと
「最も適切なものを選べ」	一番ぴったりなものを選べ	「最も」＝一番

5-2. 正誤判定問題の「チェックポイント発見法」

正誤問題は「文のどこかに1箇所だけ嘘がある」構造。嘘は具体的な名詞に仕込まれる。

嘘の仕込み場所	典型パターン	見つけ方
固有名詞の差し替え	カーナビ→ETC車載器、POP→IMAP	「このモノの名前、この文脈で正しいか？」
分類の間違い	著作者人格権に貸与権を入れる	「このグループに本当に入るか？」
動作の逆転	ダウンロード（POP）をIMAPに付ける	「この動詞、この主語で正しいか？」
数字・単位の変更	速度や容量の数値を変える	「覚えている値と一致するか？」
条件の追加・省略	「常に～できる」（実は条件付き）	「常に」「必ず」「すべて」は疑え

5-3. 穴埋め問題の「ヒント発見法」

鉄則：空欄の前後3語にヒントがある。選択肢を見る前に「どういう意味の言葉が入るか」を推測させてから選ばせる。

例1：「インターネットに接続するために，〔　〕というサービス事業者と契約する必要がある」

ヒント：「インターネット」「接続」「サービス事業者」→ **ISP（プロバイダ）**

例2：「〔　〕は，IPアドレスとドメイン名を対応させるサーバである」

ヒント：「IPアドレス」「ドメイン名」「対応させる」→ 翻訳係 → **DNSサーバ**

5-4. 「程度の表現」と正誤の傾向

正誤問題で「常に」「すべて」などの極端な表現は誤りの可能性が高い。この感覚を教える。

表現	強さ	正誤での傾向	理由
常に / 必ず / すべて	★★★ 絶対	誤りが多い	例外がないことは稀
通常 / 一般的に	★★ 普通	正しいことが多い	例外を認めている
～できる / ～可能	★ 能力	要注意	条件付きかどうか確認
～のみ / ～だけ	★★★ 限定	誤りが多い	他の方法もあることが多い
～してはならない	★★★ 禁止	法律系は正しいことが多い	法律は明確な禁止規定あり

6. 暗記技術の体系（学生に明示的に教える内容）

「記憶力が悪いんじゃない。覚え方を知らなかっただけ。」—— このメッセージを繰り返し伝える。

6-1. チャンキング（グループ化して覚える）

400語を丸暗記は不可能。カテゴリに分ければ各グループ10〜20語になる。

カテゴリ	用語数	暗記の性質	使う技術
単位・補助単位	約15語	順序で覚える	語呂合わせ（きめぎてペ / みまなぴ）
五大装置・本体構成	約20語	構造図で覚える	図を描いて関係性で記憶
入力装置	約15語	「何を入力するか」で分類	用途別グルーピング
出力装置	約10語	「何を出すか」で分類	ディスプレイ系 vs プリンタ系
補助記憶装置	約15語	容量の大きさで序列化	小→大の順で並べて覚える
インタフェース	約15語	「何を繋ぐか」で分類	接続対象別グルーピング
OS	約10語	会社名で紐づけ	MS→Windows / Apple→macOS等
ネットワーク接続方式	約10語	速度順で序列化	遅い→速いで並べる
プロトコル	約15語	「係の名前」方式	役割で覚える
サーバの種類	約10語	プロトコルと対応づけ	プロトコル表と連動
セキュリティ脅威	約15語	攻撃手口のストーリー化	物語で覚える
法律・権利	約10語	ツリー構造	知的財産権の下位分類
アプリケーションソフト	約10語	用途別	何をするソフトか
情報社会のシステム	約10語	身近な場面と対応	生活の中のIT

6-2. 間隔反復（忘却曲線に逆らう）

エビングハウスの忘却曲線：1日後には74%忘れる。「いつ復習するか」が重要。

タイミング	やること	目的
授業当日	カテゴリの用語を初回学習（10〜20語）	初回エンコーディング
翌日	Moodle小テスト10問	24時間以内の最初の復習
3日後	間違えた問題だけ再テスト	弱点の集中強化
1週間後	カテゴリ全体の復習テスト	長期記憶への定着確認
2週間後	前＋現カテゴリの混合テスト	既出概念の再利用（D-4）

6-3. アウトプット練習（「見て覚える」は最弱）

暗記の効率順：テスト効果 ＞ 説明する ＞ 書く ＞ 読む。「隠して思い出す」が最強。

練習形式	具体的なやり方	フレームワーク
フラッシュカード	用語→意味、意味→用語の双方向	D-1 演習量
穴埋めテスト	章末問題の形式そのまま	D-2 反復
仲間分け問題	同じ意味のグループに分けろ	B-3 表現揺れ
ペア合わせ	略語とその説明をマッチング	B-1 略語
比較表の空欄埋め	紛らわしいペアの片方を隠す	混同防止
自分で語呂を作る	覚えにくいものは自作の語呂で	F-3 主体的学習

6-4. セキュリティ用語はストーリーで覚える

攻撃手口を「物語」にすると、個別暗記より遥かに定着する。

攻撃ストーリー：フィッシング（偽メールで釣る）→ ソーシャルエンジニアリング（心理的に騙す）→ ショルダーハッキング（のぞき見）→ トラッシング（ゴミ箱漁り）→ なりすまし（他人のふりで侵入）→ 不正アクセス（システム侵入）

マルウェアの家族：ウイルス（寄生）→ ワーム（自力増殖）→ トロイの木馬（便利なふりして罠）→ ランサムウェア（人質に身代金）→ スパイウェア（こっそり情報窃取）→ バックドア（裏口から再侵入）

7. 計算問題の完全対策（2パターンだけ）

パターンA：補助単位の換算

「1000倍ごとに単位が変わる」だけ覚えれば解ける。べき乗の理解は不要。

覚える順番：B → KB → MB → GB → TB → PB　（×1000ずつ）

語呂：「ビケメギテペ」

例題：1枚4MBの写真1500枚 → 4×1500 = 6000MB = 6GB → DVD-R DL（8.5GB）が必要

パターンB：データ量の概算

「単位あたりのサイズ × 個数 → 単位変換」の2ステップ。

例題：1ページ4000B×500ページ → 2,000,000B = 2MB → CD-R（650MB）で十分

画像：2000×1000px × 3B/px = 6,000,000B = 6MB

8. 15週間スケジュール【統合版】

暗記レベルの低い順に配列。各週で「暗記技術」と「読解技術」を並行して導入する。

週	暗記テーマ	導入する暗記技術	読解トレーニング（冒頭10分）	Moodle
1	オリエンテーション	暗記法の全体像	「文の分解」デモ＋表現揺れ一覧配布	——
2	第5章 情報モラル①	チャンキング＋ツリー構造	パターン③：著作権の否定＋例外文	権利10問
3	第5章 情報モラル②	ストーリー化	パターン③：人格権vs財産権の比較文	脅威15問
4	第3章 アプリ	用途別グルーピング	パターン④：ソフト種類の列挙文	ソフト10問
5	第1章① 情報・単位	語呂合わせ	パターン①：JIS定義文の分解	単位10問
6	第1章② 五大装置	構造図で覚える	パターン④：装置の列挙文	装置20問
7	第1章③ 補助記憶	比較表で覚える	命令語辞典配布＋問題文の読み方	規格15問
8	第1章④ OS・パス	パス問題	パターン②：条件＋結果文（ETC等）	パス10問
9	第1章⑤ 計算＋復習	計算パターン反復	比較文の分解（シリアルvsパラレル等）	計算5＋混合20問
10	第4章 情報社会	場面との紐付け	パターン①②複合：システム定義文	システム15問
11	第2章① 接続・LAN	速度順序列化	パターン②：プロトコル定義文	接続10問
12	第2章② プロトコル	「係の名前」方式	チェックポイント発見法の実践	プロトコル15問
13	第2章③ WWW・メール	紛らわしいペア整理	「程度の表現」で正誤問題を解く	第2章25問
14	全範囲総復習	弱点の集中補強	正誤問題の「嘘の仕込み場所」総ざらい	全範囲50問
15	直前対策＋本番	満点チェックリスト	「分解してから解く」最終確認	誤答再テスト

8-1. 各週の授業構成（90分の基本パターン）【統合版】

時間	活動	柱	フレームワーク
0〜5分	前回の復習クイズ（5問）	暗記	D-2 復習の機会

5〜15分	文の分解トレーニング（テキストの長い文を1つ選んで分解）	読解	A-1 文構造対策
15〜20分	今日の暗記技術を紹介	暗記	F-4 自力努力の仕方
20〜25分	落とし穴の事前説明（偽の既知語、表現の揺れ、命令語等）	読解＋暗記	B-1,B-2,B-3 対策
25〜50分	テキスト内容の説明（1カテゴリ10〜20語）	暗記	A-3 密度管理
50〜70分	暗記技術を使った練習（グルーピング、カード等）	暗記	D-1 演習の時間確保
70〜85分	Moodle小テスト＋答え合わせ	暗記＋読解	D-1, D-2
85〜90分	まとめ＋宿題の指示（翌日Moodle復習テスト）	暗記	F-2 復習方法

旧版との違い：5〜15分の「文の分解トレーニング」が新規追加。これにより、暗記パートの説明時間を25分→25分に維持しつつ、毎回10分の読解訓練が入る。冒頭に読解を入れることで、その後のテキスト説明パートの理解度が上がる効果も期待できる。

9. Moodle問題バンク設計【統合版】

【暗記系カテゴリ】

カテゴリ	問題形式	問題数	出題のポイント
補助単位	穴埋め選択	10問	順番の問題、換算の問題
五大装置	マッチング	15問	装置名⇔役割の対応
入出力装置	マッチング	20問	装置名⇔用途の対応
補助記憶装置	正誤＋計算	15問	容量の比較、メディア選択
インタフェース	マッチング	15問	規格名⇔特徴の対応
OS・ファイル管理	穴埋め＋パス	15問	パスの指定が中心
プロトコル	マッチング	15問	略語⇔「係の名前」
サーバの種類	マッチング	10問	サーバ名⇔機能
メール	穴埋め＋正誤	15問	CC/BCC、SMTP/POP
WWW・検索	穴埋め＋正誤	10問	URL構成、検索方式
アプリケーション	マッチング	10問	ソフト名⇔用途
情報社会	穴埋め＋正誤	15問	システム名⇔場面
電子商取引	穴埋め	10問	BtoB/BtoC/CtoC
セキュリティ脅威	正誤＋マッチング	20問	攻撃名⇔手口
法律・権利	穴埋め＋正誤	15問	権利の分類、保護法

【落とし穴系カテゴリ】

カテゴリ	問題形式	問題数	出題のポイント
偽の既知語	正誤判定	10問	日常語とIT用語の意味の違い
IT用語の表現揺れ	グルーピング	10問	同一概念の表現を選ぶ
紛らわしいペア	比較問題	12問	AとBの違いを問う

【読解系カテゴリ】—— 新規追加

カテゴリ	問題形式	問題数	出題のポイント
命令語の理解	マッチング	8問	「適切な字句」「解答群」等の言い換え
正誤判定の読解	正誤（嘘を指摘）	15問	「嘘の仕込み場所」を指摘させる
穴埋めヒント抽出	穴埋め＋記述	10問	空欄の前後3語を抜き出してから選択

程度の表現	正誤判定	10問	「常に」「すべて」を含む文の正誤
文の骨格抽出	記述式	10問	長い文の「誰が・何を・どうする」を書かせる
日本語の表現揺れ	グルーピング	10問	同義の動詞群を選ぶ

合計：約345問（暗記系225問＋落とし穴系32問＋読解系63問＋模擬テスト50問）。ランダム出題＋正答率追跡でPDCAを回す。

10. 教員向け運用ガイド【統合版】

10-1.「満点」を目標にする意味の伝え方

「70点で受かった人は覚え方を振り返らない。100点を目指す人は、なぜこの問題を間違えたかを考えざるを得ない。そのプロセスが、覚え方のコツの発見になる。」

10-2. 不合格者への対応

合格率90%の試験で不合格になった場合、原因は「問題の難しさ」ではなく「覚え方の技術不足」または「問題文の読み方の不足」に特定できる。Moodleの正答率データから弱点を特定し、暗記系と読解系のどちらが原因かを切り分ける。

切り分けの目安：

・用語マッチング問題は正答するが正誤問題で失点 → **読解が原因**（文が読めていない）

・どの形式でも同じカテゴリで失点 → **暗記が原因**（用語が定着していない）

10-3. 成功体験の設計

最初の小テスト（第5章・情報モラル）で高得点を取らせることが重要。第5章は日常語に近く、常識的判断で解ける問題が多いため、成功体験の設計に最適。ここで「自分は覚えられる人間だ」という自己効力感を作る。

10-4.「読めない」ことを恥ずかしいと思わせない

学力の弱い学生は「漢字が読めない」「文の意味がわからない」ことを隠す。教員が最初に「このテキストの日本語は難しい。大人でも一読で理解できない書き方をしている。だからこそ『読み方のコツ』を教える」と宣言する。問題は学生の能力ではなく、テキストの書き方にある。

10-5.「文の分解」は国語の授業ではない

文法用語（主語・述語・連体修飾語…）は使わない。「誰が？」「何を？」「どうする？」の3つだけ。学生に「国語の勉強」だと思わせた瞬間、心のシャッターが閉まる。あくまで「試験で点を取るための技術」として教える。

10-6. 表現揺れ一覧は「辞書」として渡す

テストに出す教材ではなく、テキストを読むときの参照ツールとして渡す。「この動詞が出てきたら、一覧表で仲間を確認しろ」と使い方を教える。

10-7. この戦略の本当の目的

J検3級の合格証は副産物。本当の成果は「**覚え方のコツがわかった**」「**読み方にもコツがある**」という体験そのもの。この2つが揃って初めて、後のITパスポート、基本情報技術者試験、さらには就職活動での業務知識の習得に繋がる。入学して最初の半年で「自分は勉強ができる」という実感を作ることが、その後の学びの土台になる。

学生に繰り返し伝えるメッセージ：

「記憶力が悪いんじゃない。覚え方を知らなかっただけ。そして、読み方にもコツがある。」

ビジネスOS検定

ビジネスOS検定

商人道コースウェア設計提案書

― 「再反転」のための教育設計 ―

清風情報工科学院

2026年2月13日

Draft v0.1

第1章　なぜ商人道を教えるのか

1.1　問題の所在：大阪商人道の断絶

大阪は「商人の街」として知られるが、現在の大阪で自然に身につく商売の流儀は、本来の商人道とは正反対の厚黒学的行動様式である。本来の商人道（藤樹→梅岩→船場商人→松下幸之助）は、1945年の大阪大空襲による物理的断絶を経て、闇市→高度成長期→テレビ的イメージ固定化→自己成就的予言化という5段階の反転過程を経て事実上消滅した。

■ 反転の5段階

第1段階（幕末〜明治）：心学講舎・徒弟制度・船場言葉という慣習法的伝承の器が壊れる。精神は属人化したが消滅せず。

第2段階（1945年）：決定打。大阪大空襲で船場の商家群が壊滅。商人道は「場所と人のセット」で伝承されていたため、物理的消滅が精神的断絶を招いた。

第2.5段階（戦後〜1950年代）：焼け野原に最初に立ったのは闇市。「暖簾」「体面」「分限」は足枷となり、「面を厚く、心を黒く、素早く稼ぐ」厚黒学が最適戦略として定着。パチンコ・消費者金融で資本蓄積層が台頭。

第3段階（1950〜70年代）：高度成長の「速度」が商人道を振り落とす。丁稚制度消滅、職住分離加速。船場商人の六徳のうち前半3つ（始末・才覚・算用＝稼ぐ技術）だけ残り、後半3つ（奉公・体面・分限＝続ける精神）が脱落。

第4段階（1970年代）：テレビが厚黒学的「大阪商法」イメージを全国拡散。「細うで繁盛記」（1970年）、吉本興業の「ケチ・がめつい」芸風、「なにわ金融道」（1990年代）が決定的。

第5段階（1990年代〜現在）：最も深刻。大阪人自身が「これがうちらの伝統や」と内面化。本来の船場商人道を知らないまま、厚黒学的セルフイメージが固定化→実際にそう振る舞う人が増える→イメージ強化のループ。

1.2　数字が示す「家を滅ぼす」過程

帝国データバンクの調査によれば、大阪府は1982年以降43年連続で企業の転出超過が続いている。2025年上半期もこの傾向は変わらず、44年連続となる見込みである。これは「傾向」ではなく「構造」である。

> *“商人で道を知らない者は、ただ貪ることだけをして家を滅ぼす”* —石田梅岩『都鄙問答』

梅岩のこの警告は、400年の時を超えて都市レベルで実証されている。商人道の喪失→厚黒学の定着→都市としての信用毀損→企業流出。「ただ食ることだけをして」いれば、個人の家だけでなく、都市そのものも滅ぶ。

1.3　なぜ清風情報工科学院で教えるのか

小林一三は「大阪の空気を識らなければ大阪で仕事ができない」と語った。しかし今の大阪で自然に「空気」から吸収されるのは厚黒学のほうである。だからこそ教育課程として意識的に「本来の商人道」を注入する必要がある。

清風情報工科学院の留学生は「外部から大阪に来た人間」であり、彼らが日本で働くとき、厚黒学的イメージしか持っていなければ、それが「日本の商売のやり方」だと認識してしまう。正しい系譜を教えることで、「再反転」の担い手となりうる。

第2章　教育設計の原則

2.1　梅岩の方法論を継承する

石田梅岩は「文字芸者」（学問を飾りとする者）を痛烈に批判し、町人の言葉で町人に語りかけた。ビジネスOS検定も同じ方法論を取る。道徳の衣をかぶせた瞬間に、「稼ぎたい」人間は耳を閉じる。しかし「こっちのほうが儲かる」と言えば、目の色が変わる。入口は損得、歩いているうちに道徳になっている―これが教育設計の核心である。

■ 教材の引用順序：体感→実証→権威

1. 体感（日常の言葉）：「お上の金たよりにせんと働け。屁理屈言わんと働け。一生懸命働け」
2. 現代の実証（経営者の実例）：松下幸之助「共存共栄」、稲盛和夫「利他の経営」、身近な中小企業の実話
3. 古典の権威（梅岩・藤樹）：「これは400年前に石田梅岩が同じことを言っている」と添える

古典が先ではなく、実感が先。古典は裏付け。梅岩自身が町人に向かって「孔子がこう言った」から入ったのではなく、「あんたら毎日やってる商売、あれが道なんや」から入って、後から「これは聖人の教えと同じことや」と繋いだ。この順序を厳守する。

2.2　「稼ぎたい」を否定しない

梅岩が言ったのは「商人の売利は士の禄に同じ」―稼ぐことは武士が俸禄を得るのと同じく正当である。問題は「稼ぎたい」ではなく、「稼ぎ方を選ばない」ところにある。

教えるべきは「稼ぐな」ではなく、「稼ぎ続けられる稼ぎ方」と「一代で燃え尽きる稼ぎ方」の違い。偏差値40の学生に響くのは道徳ではなく損得勘定。だったら損得勘定で教える。

> *“三年で稼いで終わりたいんか、三十年稼ぎ続けたいんか、どっちや？”*―ビジネスOS検定・導入問

三十年と答えた瞬間に、商人道の全ての教えが「自分の利益のための合理的戦略」として入ってくる。信用は投資、体面は資産、分限はリスク管理、始末はコスト最適化。全部「稼ぐため」の技術として教えられる。

2.3　留学生への応用：「国の暖簾」

短期滞在の留学生には「三十年の信用を築け」は響かない。しかし視点を変える。「お前が帰った後、お前の後輩が来る」―これが鍵。

留学生が厚黒学で数年稼いで帰国すれば、次に来る同胞の後輩は日本人から信用されない。逆に真っ当にやれば、後輩の道が開ける。これは商人道の「暖簾」の論理そのもの。個人の暖簾ではなく「ベトナム人の暖簾」「インド人の暖簾」。船場商人が「家の名」を守ったのと同じ構造を、「国の名」「出身校の名」に置き換える。

> *“お前は個人で来ているつもりかもしれないが、お前が日本で働く姿は、お前の国の暖簾になる。その暖簾を汚すか、磨くか。”*—コースウェア設計理念

2.4　反面教師としての現代事例

厚黒学がどう破綻するかを示す反面教師は、最も強力な教材となる。武富士の武井家はその典型：ヤミ米行商からスタート→高利貸しで巨富→社員に写真への挨拶を強制→ジャーナリスト盗聴で逮捕→有罪判決→肝不全で死去（最期の言葉「家に帰りたい。連れて行け」）。息子は香港・オランダを経由した1600億円の租税回避スキームで国税と最高裁まで争い、会社は2010年に破綻、2017年に法人格消滅。金は残ったが、暖簾も体面も社会的信用も全て失われた。

第3章　本来の商人道 vs 厚黒学的「大阪商法」

観点	本来の商人道	厚黒学的「大阪商法」
利益観	利は勤むるにおいて真なり	利益は奪い取るもの
顧客観	先も立ち我も立つ	相手の損が自分の得
倹約観	始末＝効用を使い切る感謝	ケチ＝出し渋り
信用観	信用の足らざるを憂うべし	騙されるほうが悪い
社会観	陰徳善事、私財で橋を架ける	自分さえ儲かればよい
自己規律	分限・体面	面の皮の厚さが勝負

船場商人の六徳と現代ビジネス用語の対応

六徳	伝統的意味	現代ビジネス用語
始末	効用を使い切る（もったいない精神）	コスト最適化・リーン経営
才覚	商機を見抜く目利き	マーケットセンス・イノベーション
算用	数字に基づく判断	データドリブン経営・財務リテラシー
奉公	主人・顧客・社会への誠実	ステークホルダー・マネジメント
体面	信用に値する振る舞い	ブランド・レピュテーション管理
分限	身の丈を知る・過剰拡大を避ける	リスク管理・ガバナンス

高度成長期に脱落したのは後半3つ（奉公・体面・分限）。前半3つ（稼ぐ技術）だけでは「稼ぎ続ける」ことはできない。ビジネスOS検定は、この後半3つの復元を核とする。

第4章　コースウェア構成案

4.1　導入モジュール：「どっちが得や？」

対象：全受講者（日本人学生・留学生共通）

- 導入問：「三年で稼いで終わりたいんか、三十年稼ぎ続けたいんか、どっちや？」
- 対比事例：武富士（一代で燃え尽きた厚黒学）vs 鴻池家・野村家（数代続いた商人道）
- 数字で示す：大阪府43年連続企業転出超過―厚黒学が都市レベルで何を引き起こしたか
- 結論：「稼ぎたいなら長期で考えろ。長期で考えるなら信用を積め」

4.2　本論モジュール1：「商人道の系譜」

中江藤樹→石田梅岩→心学講舎→船場商人→近代経営者（野村・小林・松下）の流れを、偏差値40でも理解できる言葉で教える。

- 松下幸之助は梅岩を読まずに梅岩と同じことを言った―これが「慣習法的伝承」の力
- 小林一三の「大阪の空気を識れ」―外部者が大阪精神と融合する方法論
- 渋沢栄一『論語と算盤』―梅岩の「道徳と経済の両立」を明治の言葉で翻訳した例

4.3　本論モジュール2：「六徳を現代で使う」

六徳それぞれについて、現代のビジネス場面でどう使うかをケーススタディで教える。

- 始末：100円ショップのダイソー創業者・矢野博丈の「無駄を徹底的に省く」経営
- 才覚：小林一三の「乗客は電車が創造する」式の市場創造思考
- 算用：野村徳七の「博打を科学にする」―調査部設置の合理性
- 奉公：松下幸之助「社会貢献が使命、その報酬が利益」
- 体面：「清風の卒業生なら安心」という暖簾を自分たちが作る意識
- 分限：バブル期に無限拡大した企業の末路 vs 「身の丈経営」の堅実さ

4.4　留学生専用モジュール：「国の暖簾」

対象：外国人留学生

- 「お前の後輩が来年来る」―自分の行動が同胞の信用を左右する構造の理解
- 先輩がちゃんとやった国は次の年から求人が増えた実話、逆に逃げた国は面接すら通らなくなった実話
- 「個人の暖簾」と「国の暖簾」の二重構造―短期滞在者にも長期の視点を植え付ける唯一の回路

- 梅岩の「自分一代ではなく、家を続ける」の拡張版として位置づけ

4.5　実践モジュール：「反面教師を見分ける目」

「この街で生き残るために、周囲の商売人を真似してはいけない場面がある。何を真似すべきで、何を真似すべきでないかを見分ける目を養え」―これがビジネスOS検定の最終目標。

- 大阪の街で実際に見聞きする商行為を、六徳に照らして評価する演習
- ニュースや事件を素材にしたケースディスカッション
- 「自分ならどうする」を問う場面設定型の検定問題

付録　コースウェアで利用可能な格言・名言集

A.　中江藤樹（近江聖人、1608-1648）

> *“聖人になろうと思えばなれないはずはないのだ”*—藤樹『大学解』

→ *偏差値も学歴も関係ない。梅岩の「文字芸者批判」に繋がる根拠。*

> *“人生の目的は利得ではない。正直である、正義である”*—藤樹語録

→ *ただし教育設計上は「正直のほうが長期的に得」という形で導入する。*

> *“天地の間に、己一人生きてあると思うべし。天を師とし、神明を友とすれば外人に頼る心なし”*—藤樹語録

→ *依存しない精神。「お上の金たよりにせんと働け」の古典的根拠。*

> *“学者とは、その徳を称える呼び名だ。学識があるからそう呼ばれているのではない”*—藤樹語録

→ *知識よりも人格。検定が測るべきは暗記力ではなく判断力。*

> *“わずかな年月であらゆる知識を詰め込むのは良くない”*—藤樹の教育論

→ *藤樹自身が知的障害のある弟子・大野了佐のために平易なテキストを作り直し、根気よく教えて一人前の医者に育てた。「偏差値40の学生にこそ教える」の原点。*

B.　石田梅岩（石門心学の祖、1685-1744）

> *“商人の売利は士の禄に同じ”*—梅岩『都鄙問答』

→ *稼ぐことは正当。「稼ぎたい」を否定しない教育設計の根拠。*

> *“商人で道を知らない者は、ただ貪ることだけをして家を滅ぼす”*—梅岩『都鄙問答』

→ *コースウェアの核心命題。大阪43年連続企業転出超過が都市レベルの実証。*

> *“利は勤むるにおいて真なり”*—梅岩『都鄙問答』

→ *働いて得た利こそ正当。お上からの施しではなく自分の勤勉で得よ。*

> *“先も立ち、我も立つ”*—梅岩の商道徳

→ *取引相手も自分も共に利益を得る。「三方よし」の原型。現代のWin-Win概念。*

> *“信用の足らざるを憂うべし”*—梅岩の教え

→ *利益の不足ではなく信用の不足を心配せよ。信用＝長期投資。*

> *“倹約は吝嗇にあらず”*—梅岩の教え

→ *始末（効用を使い切る感謝）とケチ（出し渋り）は全く別物。*

> *“文字芸者になるな”*—梅岩の学問批判

→ *学問を飾りにするな。身体で分かるレベルまで削ぎ落とせ。教育方法論の核。*

C.　近代経営者の言葉

松下幸之助

> *“社会貢献が使命、その報酬が利益”*—松下経営哲学

→ *梅岩「商人の売利は士の禄に同じ」の現代版。松下は梅岩を直接読まず、船場の丁稚奉公で身体的に同じ思想を継承した（慣習法的伝承の証拠）。*

> *“共存共栄”*—松下経営理念

→ *梅岩「先も立ち我も立つ」の近代版。*

小林一三

> *“大阪の北郊に住み北区の工場に通勤では大阪が判らない。仕事場が北なら住居は南に置き、朝夕の通勤で大阪を識る頭がなくては大阪では仕事ができない”*—小林一三、高碕達之助への助言

→ *外部者が「大阪の空気を識る」ことの重要性。留学生教育に直結。*

> *“乗客は電車が創造する”*—小林一三の事業哲学

→ *「才覚」の近代的体現。需要を待つのではなく、需要を作る。*

野村徳七（二代）

> *“自己の利益よりも顧客の利益を先にす”*—野村商店の経営方針

→ *梅岩「先も立ち我も立つ」の金融版。調査部設置で「博打を科学にした」。*

稲盛和夫

> *“利他の心で経営する”*—稲盛経営哲学

→ *梅岩の「心の発明」を現代経営語で表現。京セラ・KDDIの創業実績が裏付け。*

D.　コースウェアの核心フレーズ（新規）

> *“お上の金たよりにせんと働け。屁理屈言わんと働け。一生懸命働け”*—ビジネスOS検定・三訓

→ *梅岩・藤樹以来の商人道を三行に圧縮したもの。これ以上削れない。*

> *“三年で稼いで終わりたいんか、三十年稼ぎ続けたいんか、どっちや？”*—導入モジュール・問い

→ *欲を否定せず、欲の時間軸を問う。三十年と答えた瞬間に商人道が合理的戦略になる。*

> *“この街で生き残るために、周囲の商売人を真似してはいけない場面がある。何を真似すべきで、何を真似すべきでないかを見分ける目を養え”*—コースウェア設計理念

→ *厚黒学が環境のデフォルトとなった大阪で、学生に伝えるべき核心メッセージ。*

> *“お前は個人で来ているつもりかもしれないが、お前が日本で働く姿は、お前の国の暖簾になる。その暖簾を汚すか、磨くか”*—留学生モジュール・核心命題

→ *短期滞在者にも長期の視点が生まれる唯一の回路。梅岩「家を続ける」の国際拡張版。*

— 以上 —

本文書は、2026年2月13日の対話記録に基づく設計提案ドラフトです。

清風情報工科学院　ビジネスOS検定プロジェクト

A 三層教授法の設計と運用

ビジネスOS検定　設計提案書

附属書類 A

三層教授法の設計と運用

体感 → 実証 → 権威

― 入口は損得、歩いているうちに道徳になっている ―

清風情報工科学院

2026年2月13日　Draft v0.1

第1章　三層教授法とは何か

1.1　原理：梅岩の教壇に学ぶ

石田梅岩は1729年、京都の自宅で無料の公開講座を始めた。聴衆は町人―商人、職人、丁稚、女中。学問の素養のない人間に「聖人の道」を教えようとした。彼がとった方法は、現代の教育設計理論に照らしても極めて合理的なものだった。

梅岩は「孔子がこう言った」から入らなかった。町人の日常―仕入れ、売掛、得意先まわり、奉公―の中にすでに「道」があると説き、「あんたら毎日やってる商売、あれが道なんや」という認識から出発した。そのうえで「この道は、実は聖人の教えと同じことなんや」と古典を接続した。

この順序は偶然ではない。梅岩は「文字芸者」―学問を飾りにする者―を痛烈に批判した。古典から入れば「ありがたい話」で終わり、体に入らない。体感から入れば、「自分のことや」と当事者意識が生まれ、そのあとに古典を添えれば「400年前から本物の知恵が言い続けてきたことだ」と重みが加わる。

ビジネスOS検定の三層教授法は、この梅岩の方法論を体系化したものである。

1.2　三層の定義

層	名称	機能	素材例
第1層	**体感（日常の言葉）**	当事者意識を喚起する。「自分のことだ」と思わせる	三訓、場面問題、損得比較、自分語の格言
第2層	**実証（現代の経営者の実例）**	「実際にそうなっている」と確信させる。抽象論ではない証拠を示す	松下・小林・稲盛の言行、武富士の末路、企業流出データ等
第3層	**権威（古典：梅岩・藤樹）**	「400年続いてきた本物の知恵」という重みで裏付ける	都鄙問答、翁問答、藤樹規、心学講舎の逸話

1.3　なぜこの順序でなければならないのか

■ 古典から入ると何が起きるか（禁忌パターン）

「石田梅岩は江戸時代の思想家で、都鄙問答において……」―この導入で偏差値40の学生がどうなるか。「テストに出るなら覚えるけど、自分には関係ない」。留学生なら「日本の昔話で

しょ」。古典は聞こえがよいが、聴き手の生活世界と接続しない限り、教養のパッケージとして素通りする。

梅岩自身がこの問題を看破していた。彼が「文字芸者」と呼んで最も嫌ったのは、知識を暗記して偉そうに語るが、自分の行動は変わらない人間である。古典から入る教育は、この「文字芸者」を量産する。

■ 実証（現代事例）から入ると何が起きるか（次善パターン）

「松下幸之助はこう言いました」―悪くはない。しかし松下は「すごい人」であって「自分」ではない。「あの人はすごかったからできたんでしょ」で終わる危険がある。特に偏差値40の学生にとって、「偉い経営者の話」は教壇から降りてこない。

■ 体感から入ると何が起きるか（正解パターン）

「三年で稼いで終わりたいんか、三十年稼ぎ続けたいんか、どっちや？」―この問いは偏差値に関係なく刺さる。全員が「稼ぎたい」と思っている。その「稼ぎたい」を否定せず、時間軸を問うことで、「自分の問題」として引き込む。

体感で引き込んだ後に実証を出すと、「ほんまにそうなるんや」という確信が生まれる。最後に古典を添えると、「400年前からわかってたんや」という畏敬が加わる。この三段階で、情報が知識になり、知識が信念になり、信念が行動になる。

核心原理：道徳の衣をかぶせた瞬間に、「稼ぎたい」人間は耳を閉じる。しかし「こっちのほうが儲かる」と言えば、目の色が変わる。

第2章　第1層「体感」の設計

2.1　設計原則

第1層の目的は、学習者に「これは自分の問題だ」と認識させることである。そのために、以下の原則を守る。

1. 日常語で語る。専門用語・古語は一切使わない。
2. 学習者の欲求を否定しない。「稼ぎたい」「楽したい」「早く帰りたい」―すべて出発点として受け入れる。
3. 二択で問う。答えが明らかな二択を提示し、「自分で選んだ」という感覚を持たせる。
4. 感情を動かす。データではなく物語で入る。怒り・驚き・共感―何でもいい、「へえ」を超えた感情反応を引き出す。

2.2　具体的教材

■ 導入問（全モジュール共通）

> *“三年で稼いで終わりたいんか、三十年稼ぎ続けたいんか、どっちや？”*

この問いは、欲望を否定せず時間軸を拡張させる装置。梅岩が言ったのは「欲を捨てろ」ではなく「欲の出し方を間違えるな」。短期の欲で走れば家を滅ぼし、長期の欲で走れば家が続く。欲が深い人間ほど、長期で考えたほうが最終的に得をする―これを体感させる入口。

■ 三訓（壁に貼る言葉）

> *“お上の金たよりにせんと働け。屁理屈言わんと働け。一生懸命働け。”*

梅岩・藤樹以来の商人道を、偏差値40でも留学生でも一読して理解できるレベルまで圧縮したもの。「これ以外の能書きは全部偽物」―この断言が三訓の力。壁に貼り、朝礼で唱和し、身体に叩き込むための言葉。

■ 場面問題（選択式）

学生の日常に即した場面を設定し、二択で問う。正解は明示せず、議論させてから第2層・第3層で裏付ける。

【場面例1】

あなたはコンビニでアルバイトしている。同僚が「廃棄弁当、持って帰ってもバレへんで」と言う。あなたはどうする？

A：もらう。捨てるよりもったいないし、誰も損しない。

B：もらわない。

→ →「始末」と「盗み」の境界線。梅岩「倹約は吝嗇にあらず」の体感版。

【場面例2】

あなたは転職活動中。A社は月給35万だが評判が悪い。B社は月給25万だが「うちで3年やれば次どこでも通用する」と言われた。どっちに行く？

A：A社。稼げるうちに稼ぐ。

B：B社。3年後を考える。

→ → 短期利益 vs 長期投資。信用＝資産の体感。

【場面例3】（留学生向け）

同じ国から来た先輩が、ビザの制限を超えてアルバイトしている。「入管にバレへんから大丈夫」と言う。あなたはどうする？

A：自分もやる。みんなやっている。

B：やらない。

→ →「国の暖簾」の体感版。先輩の行動が後輩の査証審査に影響する現実。

■ 損得の可視化

抽象論ではなく、具体的な金額・年数で示す。

【損得計算の例】

武富士の武井保雄：長者番付1位。最期は「家に帰りたい」。会社は消滅。

鴻池家：何代にもわたって大阪経済の柱。名前は今も残っている。

「どっちが得や？」―偏差値40の学生は、この問いに答えられる。

2.3　第1層の教壇での使い方

第1層は最初の15分で完結させる。場面問題を一つ出し、2〜3分議論させ、「答えは次で出す」と言って第2層に移る。学生の中に「問い」が残っている状態で次の層に進むことが重要。解答を出してはいけない。解答は第2層の事実が出す。

第3章　第2層「実証」の設計

3.1　設計原則

第2層の目的は、第1層で生まれた「問い」に対して「実際にそうなっている」という事実を提示し、確信を与えることである。

1. 現在進行形の事実を使う。「今この瞬間に動いているビジネスで同じ原理が生きている」ことを示す。
2. 手本と反面教師の対比で示す。片方だけでは「たまたま」に見える。対比すれば「構造」が見える。
3. 数字を使う。「43年連続転出超過」「1330億円追徴課税」―数字は感情論を超える。
4. 留学生には「今いる場所」の話をする。古典だけだと「日本の昔話」で終わる。今この大阪でどちらの道が動いているかを見せる。

3.2　実証素材の三類型

類型A：現代経営者の言行（手本）

商人道の原理が現代でも有効であることを、具体的な経営者の言葉と行動で証明する。

六徳との対応	経営者	発言・行動とその教訓
奉公	松下幸之助	「社会貢献が使命、その報酬が利益」。船場で丁稚奉公し、梅岩を読まずに梅岩と同じ結論に達した。慣習法的伝承が身体を通じて生きていた証拠。
才覚	小林一三	「乗客は電車が創造する」。需要を待つのではなく需要を作る。阪急電鉄・宝塚・東宝を興し、消費文化そのものを設計した。
算用	野村徳七（二代）	「自己の利益よりも顧客の利益を先にす」。調査部を設置し、証券取引を「博打」から「科学」に変えた。データに基づく判断の体現。
体面	稲盛和夫	「利他の心で経営する」。京セラ・KDDI創業、JAL再建。私腹を肥やさず、社会全体の利益を考える経営が信用＝ブランドを作った。

上記は既知の大経営者だが、学生との距離が遠い。そこで中小企業経営者・身近な事例も並行して収集する（ChatGPT検索プロンプトで補完予定）。「偉い人だからできた」ではなく「普通の商売人がやっている」と示すことが第2層の生命線。

類型B：反面教師（破綻事例）

厚黒学的行動を取った結果、何が起きたかを具体的に示す。「ああはなりたくない」は、「ああなりたい」以上に行動を変える。

【主要事例：武富士・武井保雄】

- 出自：ヤミ米行商 → 高利貸し
- 頂点：長者番付1位
- 社内統制：社員に自分の写真への挨拶を強制
- 犯罪：ジャーナリスト盗聴事件で逮捕 → 懲役3年執行猶予4年の有罪判決
- 最期：2006年、肝不全で死去。最後の言葉「家に帰りたい。連れて行け」
- 事後：長男が香港に移住し1600億円の資産移転 → 国税庁が1330億円追徴課税 → 最高裁で形式勝訴するも判事が「著しい不公平感」と異例の補足意見
- 法人消滅：2010年会社更生法、2017年法人格消滅

教壇での問いかけ：「武井保雄は金持ちになった。でも暖簾はあるか？　体面はあるか？　家の名は残ったか？」

金は残ったが、暖簾も体面も社会的信用も全て失われた。船場商人（鴻池家）は時代に乗り遅れたが家名と信用は守った。武富士は一代で燃え尽きた。梅岩の「家を滅ぼす」の完璧な実証。

【構造的反面教師：大阪府43年連続企業転出超過】

- 帝国データバンク調査：1982年以降、一度の例外もなく転出超過が継続
- 2024年：転入174社、転出212社（38社の超過）
- 2008-2017年の10年間：748社の転出超過
- 転出先：東京約40%、兵庫約24%

教壇での問いかけ：「43年間、一度も止まらんかった。偶然か？　仕組みの問題やろ。その仕組みの正体が、厚黒学や」

類型C：松下幸之助の特殊な位置

松下幸之助は、手本としてだけでなく「慣習法的伝承の証拠」として特別な位置を持つ。

松下は梅岩の著作を直接読んでいない。しかし船場で丁稚奉公した身体経験を通じて、梅岩と同じ結論に達した。「共存共栄」は「先も立ち我も立つ」の現代語であり、「社会貢献が使命、その報酬が利益」は「商人の売利は士の禄に同じ」の翻訳である。

これが意味するのは、商人道は「テキストを読んで学ぶもの」ではなく「環境の中で身体的に伝承されるもの」だということ。つまり慣習法（カンシュウホウ）的な伝承形態をとる。テキスト（成文法）に書いてなくても、場に生きていれば伝わる。逆に、場が壊れれば—1945年がまさにそう—テキストがあっても伝わらなくなる。

だからこそ、ビジネスOS検定は「テキスト」を作るだけでは不十分で、「場」を設計しなければならない。三層教授法は、失われた「場」を教室内に再現するための装置である。

3.3　第2層の教壇での使い方

第1層で投げた場面問題に対して、第2層では「こういう人が実際にいた」を見せる。15〜20分。手本と反面教師を必ず対で出す。

進行例：

1. 「さっきの問い、覚えてるか。三年と三十年。」
2. 「三年で稼いだ男がいる。武井保雄。長者番付1位。最後の言葉は『家に帰りたい』。会社は消えた。」
3. 「三十年、五十年で築いた男もいる。松下幸之助。名前は世界中で知られてる。」
4. 「どっちが得やった？　さっきの問いの答え、出たやろ。」

この後、具体的な数字（長者番付の順位、追徴課税額、パナソニックの時価総額等）を出して確信を補強する。

第4章　第3層「権威」の設計

4.1　設計原則

第3層の目的は、第1層と第2層で得た確信に「歴史的重み」を加えることである。「自分で感じ、事実で確認し、400年の知恵が裏付ける」―この三重の構造が、単なる「意見」を「信念」に変える。

1. 古典は「権威」として使う。「テストに出るから覚える」ではなく、「400年間生き残ってきた知恵」として尊重させる。
2. 原文は使わない。現代語に完全に訳したうえで、「原文ではこう言っている」と添える。
3. 分量は少なく。第3層は全体の2割以下。印象に残る一文を一つだけ出す。
4. 留学生には「日本だけの話ではない」と接続する。「どの国にも商人道はある。日本のものは400年前に体系化された」と伝えることで、自国の商慣行との比較を促す。

4.2　各観点に対応する古典の一文

六徳の各観点について、第3層で使う「決め台詞」を一つずつ定める。教壇では、これを第2層の事例のあとに「一言だけ」添える。

観点	決め台詞（現代語）	原典
利益観	「稼ぐことは正当。武士が給料もらうのと同じ。ただし働いて得た利に限る」	梅岩「商人の売利は士の禄に同じ」「利は勤むるにおいて真なり」
顧客観	「相手も得して、自分も得する。片方だけ得する商売は続かない」	梅岩「先も立ち我も立つ」
倹約観	「ケチと始末は違う。出し渋りは下品、使い切りは感謝」	梅岩「倹約は吝嗇にあらず」
信用観	「金が足りないことより、信用が足りないことを心配しろ」	梅岩「信用の足らざるを憂うべし」
社会観	「人間は本来美しい心を持っている。それを思い出せ」	藤樹「人間は本来誰もが美しい心を備えている」、梅岩の陰徳善事
自己規律	「道を知らない商人は、ただ食って家を潰す。400年前の警告が、今の大阪で実現している」	梅岩「商人にして道を知らざれば、ただ食るのみにして家を滅ぼす」

4.3　第3層の教壇での使い方

第2層で手本と反面教師を見せたあと、5分以内で完結させる。

進行例（利益観の場合）：

> *“さっき武富士と松下の話をした。実はな、400年前に京都のおっさんが同じこと言ってる。石田梅岩。「商人が稼ぐのは武士が給料もらうのと一緒で正当なことや。ただし、働いて得た利に限る。」―武井はどうやった？　働いて得たか？　人から奪い取ったやろ。松下は？　水道みたいに安いものを大量に届けて、世の中を便利にして稼いだ。400年前から答えは出てた。”*―授業での語り例

ポイントは三つ。

- 古典を「偉い話」として出さない。「京都のおっさん」くらいの距離感で出す。
- 第2層で見せた具体例と直接接続する。「さっきの話」の延長線上に古典を置く。
- 「400年前から答えは出てた」で締める。この一言が、古典の重みを学生の中に落とす。

4.4　留学生への応用：藤樹の教育姿勢

中江藤樹は、知的障害のあった弟子・大野了佐のために、わざわざ平易なテキストを書き直し、根気よく教え続けて、最終的に一人前の医者として独立させた。「偏差値が低い学生にこそ教える」―この姿勢自体が、藤樹の教育哲学の体現である。

留学生に対しても同じ姿勢をとる。日本語能力が十分でない学生のために、三訓は多言語に翻訳し、場面問題は母語でも議論できるようにする。しかし「内容のレベルを下げる」ことはしない。梅岩の心学講舎が文字の読めない女中にも開かれていたように、入口のハードルを下げつつ、到達点は妥協しない。

> *“聖人になろうと思えばなれないはずはないのだ”*―藤樹『大学解』

この言葉を、留学生の文脈に置き換えれば「日本で一流のビジネスパーソンになろうと思えばなれないはずはない」となる。偏差値も国籍も母語も関係ない。これが三層教授法の最深部にある信念である。

第5章　三層の統合―授業の全体設計

5.1　50分授業のモデル

時間	層	内容	教員の役割
0–10分	**第1層：体感**	場面問題を提示。二択で答えさせる。2〜3分のペア議論。「答えはこの後出す」と予告。	問いを投げる人。正解は言わない。「どっちや？」と煽る。議論が盛り上がったところで止める。
10–35分	**第2層：実証**	手本事例を語る（10分）。反面教師事例を語る（10分）。数字・写真・映像等の具体的素材を使用。「どっちが得やった？」で対比を確認。	語り手。講義型でよい。ただし質問は受ける。数字とエピソードで「事実」を突きつける。
35–45分	**第3層：権威**	対応する古典の「決め台詞」を一つだけ出す。第2層の事例と接続し、「400年前から答えは出てた」で締める。	権威の代弁者。「京都のおっさんが同じこと言ってる」の距離感。崇高にしない。
45–50分	**統合**	「今日の話、一言で言ったらどうなる？」と学生に聞く。三訓のどれかに着地させる。振り返りシート記入（任意）。	まとめ役。学生の言葉で締めさせる。教員の言葉で締めない。

5.2　やってはいけないこと（禁忌リスト）

禁忌	理由
古典から入る	学生が「自分には関係ない昔話」と認識し、耳を閉じる。梅岩が最も嫌った「文字芸者」的教育になる。
道徳を説く	「稼ぎたい」学生は道徳を聞いた瞬間に離脱する。入口は損得。「こっちのほうが儲かる」で入れ。
「稼ぎたい」を否定する	梅岩は「稼ぐな」とは言わなかった。「欲の出し方を間違えるな」と言った。欲を否定する教育は梅岩に反する。

手本だけ見せる	「あの人はすごいから」で終わる。反面教師との対比で構造が見える。必ずペアで出す。
古典を長く語る	第3層は全体の2割以下。一文だけ。長くなれば文字芸者。
留学生に「日本の伝統だから従え」と言う	従うのではなく理解する。「お前の国にもある商人道と同じ構造だ」と接続する。
「バランスを取る」議論をする	厚黒学にも良い面がある、などと言わない。商人道と厚黒学は並列ではない。43年連続転出超過が結論を出している。

5.3　三層教授法の本質

三層教授法の本質は、「知識の伝達」ではなく「判断力の涵養」にある。検定で測るべきは「梅岩が何年に生まれたか」ではなく「この場面でどちらを選ぶか、なぜか」である。

体感で問いを立て、実証で確信を得て、権威で裏付ける―この三段階を経た学生は、教室を出た後も、大阪の街で遭遇する様々な商行為を「六徳に照らして」評価できるようになる。それが「ビジネスOS」の意味である。OSは知識ではない。判断の基盤である。

> *“入口は損得、歩いているうちに道徳になっている。”*―三層教授法の設計理念

― 以上 ―

附属書類A：三層教授法の設計と運用

清風情報工科学院　ビジネスOS検定プロジェクト

B 第2層「実証」素材集

ビジネスOS検定　設計提案書

附属書類 B

第2層「実証」素材集

6観点×対比ペア　手本と反面教師

清風情報工科学院　2026年2月13日　Draft v0.1

序　本書の位置づけと使い方

本書は「附属書類A：三層教授法の設計と運用」の第3章（第2層「実証」の設計）で述べた実証素材を、授業で即座に使える形に整理したものである。

6つの観点それぞれについて、商人道サイドの手本と厚黒学サイドの反面教師を1対1で対比する「対比ペア」を構成した。各ペアには、授業で使う問いかけを付した。

■ 使い方

- 第1層（体感）の場面問題で学生に問いを投げた後、第2層としてこのペアを提示する。
- 手本→反面教師の順で語り、最後に問いかけで締める。所要時間は1ペアあたり15〜20分。
- ペアは独立しているので、授業の観点に応じて任意の順序で使える。
- 「教材として最も強力な3ペア」は本書末尾に示す。初回授業にはこの3ペアを推奨。

■ 出典について

各事例の出典URLは本書末尾の「出典一覧」にまとめた。授業で使う際には出典の明示は必須ではないが、学生から質問があった場合に参照できるようにしておくこと。なお、事例は全て結末が確定しているもの（刑事判決確定、行政処分確定、倒産・清算完了等）に限定した。

第1章　対比ペア集

1.1　利益観

商人道：利は勤むるにおいて真なり　　**厚黒学：**利益は奪い取るもの

○ 手本	岡藤正広　伊藤忠商事
エピソード	「マーケットイン　利は川下にあり」を経営方針として掲げ、利益を「自分が取りに行く」のではなく、顧客・市場の価値創出から生まれるものと位置づけた。川上の都合ではなく川下（消費者・現場）に合わせて商売を組み立てることで、利益の正当性を担保するという思想。2025年の新入社員向けメッセージでも「利益よりも信用を失うことのないように」と念押ししている。
結果	非財閥系で五大商社首位級の純利益。不正による崩壊事例なし。
教訓	**利益は「奪う」より「価値から生まれる」に寄せるほうが長く続く。**

✕ 反面教師	菊川剛（元会長）　オリンパス
厚黒学的行動	金融商品の損失を隠すために粉飾決算を行い、利益や財務状態を実態より良く見せかけた。経営トップ級の関与が裁判で認定され、元会長に有罪判決（執行猶予付き）。外国人社長（ウッドフォード氏）が内部告発したが解任されるという、隠蔽体質の深刻さも露呈した。
結末	元会長に有罪判決。会社にも罰金刑。株価暴落、長期の信用毀損。
教訓	**「数字で勝つ」は短期的。発覚した瞬間に全部失う。**

授業での問いかけ：

> *“同じ上場企業でも、「市場の価値から利益を得る」のと「数字を作って利益を見せかける」のと、どちらが30年後に信頼されている？”*

1.2　顧客観

商人道：先も立ち我も立つ　　**厚黒学：**相手の損が自分の得

○ 手本	長谷川喜昭　ツカサ（インテリア流通）
エピソード	「メーカー様とお得意先様の両者に喜んでいただくことが、ひいては当社の繁栄にもつながる」と明言。取引関係をゼロサムではなく両立として設計し、短期の搾取より長期の関係維持を選ぶ姿勢を経営方針として公開している。流通業という中間に立つ立場で、上流にも下流にも利益を残す商売の仕方を示す。
結果	企業として存続。制裁・不祥事なし。
教訓	**相手が立たない商売は、いずれ自分も立たない。**

✕ 反面教師	（法人としてのアマゾンジャパン）　アマゾンジャパン合同会社
厚黒学的行動	取引上弱い立場にある納入業者に対し、交渉なく代金を減額する等の行為が「優越的地位の濫用」として問題になった。公正取引委員会の手続きにより確約計画の認定を受け、是正・回復措置が求められた（2020年）。強者の論理で一方的に利益を確保しようとした構造。
結末	公取委による確約計画認定。社会的批判と取引慣行の是正を迫られた。
教訓	**「強い立場の押し切り」は、規制と不信で返ってくる。**

授業での問いかけ：

> *“同じ「取引」でも、共存共栄と優越的地位の押し切り。どっちが長期に安定する？　お前が独立して商売するとき、どっちの会社と取引したい？”*

1.3　倹約観

商人道：始末＝効用を使い切る感謝　　**厚黒学：**ケチ＝出し渋り

○ 手本	堀埜一成（元社長）　サイゼリヤ
エピソード	「『究極のおいしさ』よりも『安定したおいしさ』を優先」と語り、過剰品質・過剰サービスを追わず必要十分を徹底。掃除機がけの廃止など「必要性から工程を削る」オペレーション改善を積み重ねた。削ったのは「無駄」であって「必要」ではない。この区別が始末の本質。低価格で安定した食事を提供し続けるビジネスモデルの根幹。
結果	低価格路線で安定成長を継続。外食産業のコスト効率の手本として評価。
教訓	**「削るべき無駄」と「残すべき必要」を見極めるのが倹約。間違えると事故になる。**

✕ 反面教師	南谷昌二郎（当時会長）ほか　JR西日本
厚黒学的行動	2005年福知山線脱線事故。107名が死亡。安全管理体制・組織運営のあり方が社会的に厳しく問われた。ダイヤ優先・効率優先の経営姿勢が安全に必要な投資と余裕を削いだとの批判。「削るべきでないもの」を削った結果としての最悪のケース。事故後、経営首脳の引責退任が報道・確定。
結末	死者107名。経営首脳の引責退任。長期にわたる社会的信用毀損と遺族対応。
教訓	**安全・品質を「出し渋る」と、金では取り返せないコストを払う。**

授業での問いかけ：

> *“同じ「コスト意識」でも、無駄を削る倹約と、必要まで削る出し渋り。線を引く場所を間違えたら何が起きる？　サイゼリヤが削ったのは掃除機がけ。JR西日本が削ったのは安全の余裕。この差が107人の命の差になった。”*

1.4　信用観　★推奨ペア第1位

商人道：信用の足らざるを憂うべし　　**厚黒学：**騙されるほうが悪い

○ 手本	豊田章男　トヨタ自動車
エピソード	2010年、大規模リコール問題で米国議会の公聴会に出席。「安全について不安を感じており、それはひとえに私の責任」と述べ、品質問題を現場のせいにせず、トップの責任として前面に立った。批判は激しかったが、「逃げずに引き受ける」姿勢が長期的には信用の回復に繋がった。世界最大の自動車メーカーの社長が一人で議会に立つ―この姿が「信用は行動で守る」の象徴。
結果	短期的には株価下落・批判。しかし企業としては立て直し、世界首位を維持。
教訓	**信用は「守る宣言」より「責任を引き受ける行動」で守れる。**

✕ 反面教師	杉山武史（当時社長）　三菱電機
厚黒学的行動	鉄道向け空調装置等の製造工場で長年にわたる検査不正が発覚（2021年）。出荷検査のデータを改ざんし、契約通りの品質検査を行っていなかった。製造業の根幹である「検査の信頼」が崩壊。「バレなければいい」が組織文化として定着していたことが問題視された。社長が引責辞任を表明。
結末	社長引責辞任。顧客企業（鉄道会社等）からの信頼毀損。複数工場で類似不正が次々発覚。
教訓	**「バレなければいい」は、バレた瞬間に会社の根が腐る。**

授業での問いかけ：

> *“同じ製造業。品質問題が起きたとき、トップが「俺の責任だ」と議会に立ったトヨタと、検査を偽り続けた三菱電機。どちらが世界で信用され続けている？信用は金で買えるか？　買えないなら、何で守る？”*

1.5　社会観　★推奨ペア第3位

商人道：陰徳善事、私財で橋を架ける　　**厚黒学：**自分さえ儲かればよい

○ 手本	山田邦雄　ロート製薬（大阪）
エピソード	私財を投じて地域のための「山田スイミングクラブ」を設立。派手なCSR広告ではなく、地域に実際の施設と機会を残すタイプの貢献。「陰徳」の現代版―見返りとしてのPR効果を目的とせず、自分の金で自分の地域に投資する。船場商人が私財で橋を架けたのと同じ構造。大阪の製薬会社の社長が大阪の子どもたちに場を作る。
結果	地域での信頼蓄積。企業イメージの安定。制裁事例なし。
教訓	**社会に残る「実物の貢献」は、広告費より長く効く信用の土台になる。**

✕ 反面教師	（経営陣）　雪印食品（雪印グループ）
厚黒学的行動	2002年、BSE対策の国の買い取り制度を悪用し、輸入牛肉を国産と偽って補助金を詐取した（牛肉偽装事件）。社会的制度＝「公共の財布」を騙して利益を得ようとした行為。消費者と国の信頼を踏み台にした。国民全体を欺く構造であり、「自分さえ儲かればよい」の極致。
結末	経営再建を断念し、会社清算（解散）。雪印ブランド全体への信用毀損は雪印乳業にも波及。
教訓	**社会をだます利益は「会社そのもの」を消す。**

授業での問いかけ：

> *“社会に「与える」会社と、社会を「利用してだます」会社。片方は今もある。片方は消えた。お前が経営者になったとき、どっちの道を歩く？”*

1.6　自己規律　★推奨ペア第2位

商人道：分限・体面　　**厚黒学：**面の皮の厚さが勝負

○ 手本	佐藤肇　スター精密
エピソード	「『会社の存亡をかけるような大決断をしないで済ませる』本来、これが一番良い経営だ」と明言。派手な大勝負を避け、倒れない運営を優先する「身の丈経営」の思想を言葉にした。拡大の誘惑に抗い、分限（身の丈）を守ることが、長期存続の技術であるという考え方。これは梅岩の「分限」そのもの。
結果	極端な破綻事例なし。堅実経営で存続。
教訓	**勝負を減らすのは弱さではない。長生きの技術。**

✕ 反面教師	水島廣雄（会長 当時）　そごう
厚黒学的行動	土地神話の時代に百貨店の拡大路線を推進。バブル崩壊後に巨額の負債を抱え、経営が完全に行き詰まった。約1兆8700億円という戦後最大級の負債で2000年に倒産。身の丈を超えた拡大は、景気の上り坂では「有能な経営」に見え、下り坂に入った瞬間に致命傷になる。
結末	約1兆8700億円の負債で倒産（2000年）。戦後最大級の小売業破綻。
教訓	**身の丈を超えた拡大は、景気の波で折れる。**

授業での問いかけ：

> *“「大勝負で一気に勝つ」と「勝負を小さくして生き残る」。そごうは一兆八千億の大勝負で消えた。スター精密は「大決断をしないのが良い経営」と言って残っている。30年後に立っているのはどっちや？”*

第2章　初回授業の推奨ペア

6ペアの中から、「偏差値40の学生と留学生の両方に一発で刺さる」基準で3ペアを選定した。初回授業にはこの3ペアを推奨する。

第1位：信用観（1.4節）

トヨタ（豊田章男） vs 三菱電機（検査不正・杉山社長辞任）

- 「製造業・品質・検査」は留学生にも直感的に理解できる。特にインド・ベトナムからの留学生は製造業への就職を視野に入れており、自分事として聞ける。
- 「責任を引き受ける／ごまかす」の二択が明確で、結末（信頼維持 vs 信用失墜・辞任）がわかりやすい。
- 第3層への接続：梅岩「信用の足らざるを憂うべし」が一言で刺さる場面。

第2位：自己規律（1.6節）

スター精密（佐藤肇） vs そごう（水島廣雄・倒産）

- 「身の丈」と「過剰拡大」は業界知識がなくても理解できる。金額（1兆8700億円）のインパクトが大きい。
- 「派手な勝負はかっこいいが、倒れる」―この構図は「稼ぎたい」学生に最も刺さる。稼ぎ方の問題であって、稼ぐこと自体の否定ではないから。
- 第3層への接続：梅岩の「分限」＝身の丈を知ることが長期の利益になるという論理。

第3位：社会観（1.5節）

ロート製薬（山田邦雄） vs 雪印食品（牛肉偽装・清算）

- 食品偽装は国籍を問わず理解でき、「会社が消える」という結末が圧倒的に強い。
- 「社会に残す／社会をだます」の対比が一発で伝わる。特に「国の制度を悪用して金を取る」という構造は、留学生にとってビザ制度の不正利用と地続きの話として聞ける。
- 第3層への接続：船場商人の「私財で橋を架ける」陰徳善事の伝統。大阪の企業（ロート製薬）が大阪の子どもに場を作る話は、「清風情報工科学院が留学生に場を作る」と直結する。

第3章　素材の補完方針

本書の対比ペアは、公開情報で事実関係と結末が確認できるものに限定した。今後、以下の方向で素材を補完することで、教材の厚みと説得力が増す。

■ 補完方針1：大阪・関西比率の引き上げ

現在のペアで大阪・関西に直接関係するのは、観点5のロート製薬のみ。清風の学生にとっては「自分が今いる街の話」が最も刺さるため、以下を優先的に探索する。

- 商人道サイド：大阪・関西の中堅企業で、六徳的経営を実践している経営者
- 厚黒学サイド：大阪・関西で破綻・制裁が確定した企業（特に観点2「優越的地位濫用」で公取委処分を受けた関西企業）

■ 補完方針2：中小企業事例の増強

現在のペアは大企業中心。学生にとっては「自分が就職しうる規模」の会社の事例がより身近である。中小企業経営者の言行録・インタビュー・講演録を当たり、特に以下を探索する。

- 観点1（利益観）：「補助金に頼らない」を明言した中小企業経営者
- 観点3（倹約観）：「もったいない」精神で無駄を排した町工場の事例

■ 補完方針3：留学生の先輩事例

最も強力な補完は、清風情報工科学院の卒業生自身の事例である。「先輩がちゃんとやった国は、次の年から求人が増えた」「先輩が逃げた国は、翌年から面接すら通らなくなった」―こうした実話を、個人が特定されない形で収集し、「国の暖簾」モジュールの実証素材とする。これは外部検索では得られない、清風固有の教材資産となる。

出典一覧

商人道サイド

[S-01] 伊藤忠商事 2025年新入社員メッセージ: https://www.itochu.co.jp/ja/news/press/2025/250401.html

[S-02] 伊藤忠 岡藤会長インタビュー（TBS NEWS DIG）: https://newsdig.tbs.co.jp/articles/withbloomberg/879082

[S-03] ツカサ 社長メッセージ: https://ti-tsukasa.co.jp/recruit/message/

[S-04] サイゼリヤ 堀埜一成 PRESIDENT Online: https://president.jp/articles/-/100923

[S-05] サイゼリヤ 堀埜一成 おなじみ: https://onaji.me/entry/2026/02/06

[S-06] 豊田章男 公聴会全文（Response.jp）: https://response.jp/article/2010/02/24/136844.html

[S-07] 豊田章男 リコール会見（Bloomberg）: https://www.bloomberg.co.jp/news/articles/2010-02-05/KXD9750D9L3601

[S-08] ロート製薬 山田邦雄: https://www.osaka21.or.jp/publishing/e-book/no138/138_3.pdf

[S-09] スター精密 佐藤肇: https://plus.jmca.jp/aidoku/aidoku025.html

厚黒学サイド

[A-01] オリンパス粉飾決算（ITmedia）: https://www.itmedia.co.jp/news/articles/1307/03/news080.html

[A-02] ライブドア事件（corporate-legal.jp）: https://www.corporate-legal.jp/news/543

[A-03] 東芝不適切会計（東洋経済）: https://toyokeizai.net/articles/-/78801

[A-04] スルガ銀行 業務停止命令（金融庁）: https://www.fsa.go.jp/news/30/ginkou/20181005/20181005.html

[A-05] アマゾンジャパン 確約計画認定（公取委）: https://www.jftc.go.jp/houdou/pressrelease/2020/sep/200910.html

[A-06] 三菱自動車 元社長有罪判決（Response.jp）: https://response.jp/article/2008/01/17/104356.html

[A-07] 雪印集団食中毒事件（Wikipedia）：https://ja.wikipedia.org/wiki/雪印集団食中毒事件

[A-08] 雪印牛肉偽装事件（Wikipedia）：https://ja.wikipedia.org/wiki/雪印牛肉偽装事件

[A-09] 三菱電機 検査不正・社長辞任（YouTube/テレ朝）：https://news.tv-asahi.co.jp/news_economy/articles/000122276.html

[A-10] 神戸製鋼所 データ改ざん・会長辞任（テレ朝）：https://news.tv-asahi.co.jp/news_economy/articles/000122276.html

[A-11] そごう倒産・水島廣雄（Wikipedia）：https://ja.wikipedia.org/wiki/水島廣雄

[A-12] タカタ 民事再生法申請（Reuters）：https://jp.reuters.com/article/world/17-idUSKBN19H01F/

— 以上 —

附属書類B：第2層「実証」素材集

清風情報工科学院　ビジネスOS検定プロジェクト

C 第2層「実証」素材集──IT企業経営者版

ビジネスOS検定　設計提案書

附属書類 C

第2層「実証」素材集——IT企業経営者版

6観点×対比ペア　手本と反面教師

清風情報工科学院　2026年2月13日　Draft v0.1

序　本書の位置づけと使い方

本書は「附属書類B：第2層実証素材集」のIT企業経営者版である。附属書類Bが製造業・流通業・金融業を中心に構成しているのに対し、本書は1990年以降（主に2000年以降）のIT企業に限定して対比ペアを構成した。

清風情報工科学院の学生はIT業界への就職を目指しており、IT企業の事例は「自分が行く世界の話」として最も身近に響く。附属書類Bと本書を併用することで、第2層の実証素材を業種横断的に展開できる。

■ 使い方

基本的な使い方は附属書類Bと同じである。第1層（体感）の場面問題で学生に問いを投げた後、第2層としてこのペアを提示する。手本→反面教師の順で語り、最後に問いかけで締める。所要時間は1ペアあたり15〜20分。

附属書類Bのペアと本書のペアは同じ6観点で構成しているため、授業の構成に応じて混在させることもできる。たとえば「信用観」を2回扱う場合に、1回目にBのトヨタvs三菱電機、2回目にCのApple vs Metaを使う、という運用が可能。

■ 附属書類Bとの関係

本書は附属書類Bの「代替」ではなく「補完」である。附属書類Bの日本企業中心のペアは、「日本で働くための基本」として不可欠。本書のIT企業ペアは、「IT業界で働くための応用」として位置づける。

■ 出典について

各事例の出典URLは本書末尾の「出典一覧」にまとめた。事例は全て結末が確定しているもの（刑事判決確定、行政処分確定、破産手続完了等）に限定した。

第1章　対比ペア集

1.1　利益観

商人道：利は勤むるにおいて真なり　　**厚黒学：**利益は奪い取るもの

○ 手本　　Jason Fried　Basecamp	
エピソード	「ベンチャーキャピタルは助けるより殺すことが多い」と語り、資金で急拡大するより、実業として回る形（小さく・遅く成長）を選ぶ思想を明確にしている。外部資金で'見せかけの成長'を作るのではなく、顧客に価値を出して売上で立つという方向。IT業界では「調達額＝成功」という空気が強い中、それに抗う姿勢を貫いている。
結果	小さくても継続・存続。VC依存の破綻型ではなく、利益で回り続けるモデル。
教訓	**稼ぎは'資金調達'より'仕事（価値）'で作る。**

✕ 反面教師　　Sam Bankman-Fried（創業者CEO）　FTX（暗号資産取引所）	
厚黒学的行動	暗号資産取引所FTXは、顧客資金の不正流用を含む複数の詐欺スキームで急成長したとされ、刑事で有罪・重刑に至った。ビジネスの実態より「資金を動かす力」で利益を作ろうとして崩壊。創業者は「有効な利他主義」を掲げていたが、その資金源は顧客の金だった。
結末	2022年に破産。創業者は禁錮25年の判決確定（米司法省、2024年）。
教訓	**"奪う利益"は、最後に全部を奪われる。**

授業での問いかけ：

> *BasecampとFTX、どちらも"儲け方"を選んだ。価値で稼ぐのと資金操作で稼ぐのは何が違い、どちらが最後に残ったか？*

1.2　顧客観

商人道：先も立ち我も立つ　　**厚黒学：**相手の損が自分の得

○ 手本　　Jeff Bezos（Amazonの原則として確立）　Amazon	
エピソード	Amazonは「Customer Obsession（顧客への執着）」「信頼を獲得し続ける」「長期で考える」をリーダーシップ原則として明文化。短期の搾取では

	なく、顧客が得をする形を積み重ねることで自社も伸びる、という方向の設計。14項目の原則の筆頭にCustomer Obsessionを置いている点が象徴的。
結果	世界的に存続・成長。顧客中心の設計が長期の競争優位に。
教訓	**顧客が得する仕組みを作ると、自分も得をする。**

✕ 反面教師　Daniel Zhang（張勇：当時CEO）　Alibaba（中国EC/プラットフォーム）	
厚黒学的行動	アリババは出店者に対する排他的取引（いわゆる「二者択一」＝自社プラットフォームだけを使え）などが独禁当局に問題視された。強い立場を使って相手の選択肢を狭めれば短期的には有利でも、規制・反発で跳ね返ってくる構造。プラットフォーム支配力の私的濫用の典型。
結末	中国当局が反独占法違反で約182億元（約2,750億円相当）の制裁金（2021年）。
教訓	**"相手を縛る得"は、制度と不信で回収される。**

授業での問いかけ：

> *AmazonとAlibaba、同じ"プラットフォーム（市場）"で、顧客中心と排他で支配。どちらが長期で強いモデルか？*

1.3　倹約観

商人道：始末＝無駄を出さない　　**厚黒学：**ケチ＝必要投資の出し渋り

○ 手本　Jeff Bezos（Amazonの原則）　Amazon	
エピソード	Amazonは「Frugality（倹約）」を原則にし、「少ない資源でより多くを成し遂げる」「予算や固定費が増えても偉くない」と明文化。創業期にはドアに脚をつけて机にした'ドア机'の逸話があり、無駄を削って本質に集中する文化が見える。倹約を「貧乏」ではなく「集中」の道具にしている。
結果	倹約を文化にしつつ、世界最大級のIT企業に成長・存続。
教訓	**倹約＝"無駄を削って本質に出す"。**

✕ 反面教師　Richard Smith（当時CEO）　Equifax（米信用情報/IT）

厚黒学的行動	Equifaxは2017年に約1億4,700万人分の個人情報が漏えいする大規模事案を起こした。基本的な脆弱性パッチの適用不備など、セキュリティ投資の管理不足が議会報告等で指摘された。'コスト削減'ではなく'必要な安全投資の出し渋り・管理の甘さ'が致命傷になった典型。
結末	大規模信用失墜。CEO退任。米FTCと最大7億ドルの和解。
教訓	**安全・品質の出し渋りは、後で何倍も払う。**

授業での問いかけ：

> *AmazonとEquifax、同じ"コスト意識"でも、無駄を削るのと必要を削るのは別物。境界線はどこか？*

1.4　信用観

商人道：信用の足らざるを憂うべし　　**厚黒学：**騙されるほうが悪い

○ 手本　　Tim Cook　Apple	
エピソード	Cookはプライバシーを「基本的人権」と位置づける発言を繰り返し、短期収益よりユーザー保護を価値として前面に出している。FBI からのiPhoneロック解除要請にも応じなかった事例は象徴的。ITの信用は'機能'ではなく'守る姿勢'で決まるという教材にしやすい。
結果	ブランド信頼の核としてプライバシーを軸に存続・成長。
教訓	**信頼は"商品"であり、長期の資産。**

✕ 反面教師　　Mark Zuckerberg　Facebook/Meta（SNS）	
厚黒学的行動	Cambridge Analytica問題等を背景に、ユーザーデータの扱いが大規模に問題化。FTCから50億ドルの制裁金と厳格な是正枠組みを科された（2019年）。ユーザーデータを「収益の道具」として軽く扱った姿勢が「信用の崩壊」として跳ね返った例。
結末	FTCから50億ドルの制裁金＋継続的監督。信用面で大きな打撃。
教訓	**「データは儲かる」より「データは信頼」――ここを外すと終わる。**

授業での問いかけ：

> *AppleとMeta、どちらもユーザー情報を扱うIT企業。守って信頼を取るか、使って制裁を受けるか。長期で得なのはどっち？*

1.5　社会観

商人道：陰徳善事　　**厚黒学：**自分さえ儲かればよい

○ 手本　　Marc Benioff　Salesforce	
エピソード	Salesforceは「1-1-1モデル」（株式の1%・製品の1%・社員時間の1%をコミュニティへ還元）を創業時から制度化。単発のCSRイベントではなく、'企業の設計図に社会貢献を組み込む'という発想。利益と社会貢献は二律背反ではないという前提で会社を設計している点がポイント。
結果	大企業として存続・成長。1-1-1モデルは多くの企業に影響。
教訓	**社会への貢献は"イベント"でなく"設計"にする。**

✕ 反面教師　　Changpeng Zhao（CZ：創業者CEO）　Binance（暗号資産取引所）	
厚黒学的行動	マネーロンダリング対策（AML）の不備等により犯罪利用を許したとして刑事責任に至った。会社は40億ドル超の解決金、創業者本人も有罪を認め禁錮刑。社会の安全や法秩序を軽視した"儲け優先"が、国家レベルの制裁として返ってきた例。
結末	会社は40億ドル超の解決金。創業者CZは禁錮4か月（米司法省、2024年）。
教訓	**社会ルールを踏む利益は、最後に利益を吹き飛ばす。**

授業での問いかけ：

> *SalesforceとBinance、どちらも"世界でお金が動く"IT企業。社会に返す設計と社会ルール軽視、どちらが持続する？*

1.6　自己規律

商人道：分限・体面　　**厚黒学：**面の皮の厚さが勝負

○ 手本　　Jason Fried　Basecamp	
エピソード	「Stay Small, Grow Slow」を掲げ、拡大そのものを目的にしない。資金を入れて急拡大する誘惑を避け、運営可能なサイズで継続する自己規律を選ぶ。IT業界では「大きくなれ」が正義とされる中、小さく保つことを戦略的に選択。これは梅岩の「分限」そのもの。
結果	小さくても20年以上存続し続けるモデル。

教訓	**"勝負を小さく保つ"のは長生きの技術。**

✕ 反面教師　Adam Neumann（創業者CEO）　WeWork（シェアオフィス）	
厚黒学的行動	不動産賃貸業をテック企業と見せかけ、ガバナンス不全・過剰拡大・自己取引疑惑がIPO失敗の引き金に。企業価値470億ドルと言われていたが、統治と足腰が追いつかないまま膨張し、一気に崩壊。急成長の「物語」を売って、中身が追いつかなかった典型。
結末	2023年にChapter 11（破産手続）申請。企業価値は急落。
教訓	**拡大は"統治"が伴わないと崩壊する。**

授業での問いかけ：

> *BasecampとWeWork、どちらも"会社のサイズ"を選んだ。小さく遅くと大きく速く、30年後に残るのはどっち？*

第2章　初回授業の推奨ペア

6ペアの中から、「偏差値40の学生と留学生の両方に一発で刺さる」基準で3ペアを選定した。IT専門科目との連動を重視し、附属書類Bとは異なる優先順位をつけている。

第1位：自己規律：Basecamp（Jason Fried） vs WeWork（Adam Neumann・破産）

・「小さく続ける」vs「急拡大で破産」が直感でわかる。ビジネス用語が少なくても理解でき、結末（破産）が派手で記憶に残る。

・IT業界で「大きくなれ」が正義とされる空気の中で「小さく保つ」という選択があることを示せる。留学生にとっても「身の丈」は母国語に翻訳しやすい概念。

・第3層への接続：梅岩の「分限」＝身の丈を知ることが長期の利益になるという論理。附属書類Bのスター精密vsそごうと同じ構造をIT業界で再現できる。

第2位：顧客観：Amazon（Jeff Bezos） vs Alibaba（張勇・独禁制裁）

・同じ巨大プラットフォームで、顧客中心と排他で支配→独禁制裁の対比が鮮明。金額（約2,750億円相当）のインパクトが大きい。

・留学生にも「市場を支配しすぎると国が止める」が伝わりやすい。中国出身の学生にはAlibaba、インド出身の学生にはAmazon Indiaの文脈で自分事として聞ける。

・第3層への接続：「先も立ち我も立つ」がプラットフォーム経済でも有効であることの実証。

第3位：信用観：Apple（Tim Cook） vs Meta（Zuckerberg・FTC制裁50億ドル）

・どの国の学生にも通じるテーマがプライバシー。iPhoneとInstagram/Facebookは学生全員が使っている。

・「守る」か「軽視して制裁」かの二択で、授業の討論が回しやすい。IT業界志望の学生にとっては、データの扱いが自分の将来の仕事に直結する話。

・第3層への接続：「信用の足らざるを憂うべし」。デジタル時代の「信用」はデータの扱い方に表れるという展開。

第3章　附属書類Bとの対照表

本書（C）と附属書類B の対比ペアを並べると、同じ6観点を異なる業界で照射していることが一覧できる。

観点	附属書類B（日本企業中心）	附属書類C（IT企業版）
利益観	伊藤忠（岡藤）vs オリンパス（菊川）	Basecamp（Fried）vs FTX（Bankman-Fried）
顧客観	ツカサ（長谷川）vs アマゾンジャパン	Amazon（Bezos）vs Alibaba（張勇）
倹約観	サイゼリヤ（堀埜）vs JR西日本	Amazon（Bezos）vs Equifax（Smith）
信用観	トヨタ（豊田章男）vs 三菱電機	Apple（Cook）vs Meta（Zuckerberg）
社会観	ロート製薬（山田）vs 雪印食品	Salesforce（Benioff）vs Binance（CZ）
自己規律	スター精密（佐藤）vs そごう	Basecamp（Fried）vs WeWork（Neumann）

授業設計のポイント：同じ観点を異なる業界で2回扱うことで、「業界を問わず通じる原則」であることが学生に伝わる。1回目にBの日本企業ペア、2回目にCのIT企業ペアを使うと、「日本の伝統→世界のIT」という流れで第3層（古典）への橋渡しが自然になる。

第4章　素材の補完方針

本書のIT企業ペアは、グローバルIT大手を中心に構成した。今後、以下の方向で素材を補完することで、清風の学生により身近な教材となる。

■ 補完方針1：日本のIT企業比率の引き上げ

現在のペアで日本のIT企業は登場しない。日本のIT上場企業（特に関西圏のSaaS・EC・Fintech・ゲーム）の創業者インタビューから、六徳的経営の実例を探索する。商人道サイドの手本として、GMOインターネットグループ（熊谷正寿）、サイバーエージェント（藤田晋）、メルカリ（山田進太郎）等の一次発言は調査済みだが、対比ペアとして組み立てるには反面教師側の事例との対応関係を精査する必要がある。

■ 補完方針2：関西のITスタートアップ事例

清風の学生にとっては「大阪のIT会社」の事例が最も刺さる。大阪・関西発のIT企業で、六徳的経営を実践している経営者の言行録を探索する。規模が小さくても「自分が就職しうる会社」の事例のほうが動機づけに効く。

■ 補完方針3：IT業界の「国の暖簾」事例

IT業界でも「信用が国籍に紐づく」事例は存在する。インドのIT人材がシリコンバレーで築いた信頼、逆にデータ不正や品質偽装で特定国からの案件が減った事例など、「国の暖簾」モジュールに接続可能なIT業界固有の実証素材を収集する。

出典一覧

商人道サイド（手本）

[T-01] Amazon Leadership Principles: https://www.aboutamazon.com/about-us/leadership-principles

[T-02] Basecamp CEO Jason Fried (Vox): https://www.vox.com/2019/1/23/18193685/venture-capital-money-kills-business-basecamp-ceo-jason-fried

[T-03] Jason Fried: Stay Small, Grow Slow (Inc.com): https://www.inc.com/will-yakowicz/jason-fried-basecamp-stay-small-grow-slow.html

[T-04] Jeff Bezos door desk (Business Insider): https://www.businessinsider.com/billionaire-jeff-bezos-amazon-uses-door-desk-from-early-days-2024-1

[T-05] Tim Cook: Privacy is a fundamental human right (IAPP): https://iapp.org/news/a/apples-tim-cook-protecting-privacy-most-essential-battle-of-our-time

[T-06] Salesforce 1-1-1 Model: https://www.salesforce.com/news/stories/how-far-can-the-1-1-1-model-go-this-tech-darling-has-a-unique-approach/

[T-07] Satya Nadella on Trust (Fortune): https://fortune.com/2024/05/20/satya-nadella-microsoft-build-generative-ai-openai/

[T-08] Satya Nadella Annual Letter (Times of India): https://timesofindia.indiatimes.com/technology/tech-news/microsoft-ceo-satya-nadella-in-his-annual-letter-to-shareholders-colleagues-and-customers-fifty-years-after-our-founding-microsoft-is-/articleshow/124800819.cms

厚黒学サイド（反面教師）

[A-01] FTX: Sam Bankman-Fried禁錮25年 (米司法省): https://www.justice.gov/archives/opa/pr/samuel-bankman-fried-sentenced-25-years-his-orchestration-multiple-fraudulent-schemes

[A-02] Theranos: Elizabeth Holmes有罪 (米司法省): https://www.justice.gov/usao-ndca/us-v-elizabeth-holmes-et-al

[A-03] Alibaba 反独占制裁 (Reuters): https://www.reuters.com/business/retail-consumer/china-regulators-fine-alibaba-275-bln-anti-monopoly-violations-2021-04-10/

[A-04] Google 比較ショッピング EU制裁 (CJEU): https://curia.europa.eu/jcms/upload/docs/application/pdf/2024-09/cp240135en.pdf

[A-05] Equifax CEO退任 (Guardian): https://www.theguardian.com/business/2017/sep/26/equifax-boss-richard-smith-retires-data-breach

[A-06] Uber $148M和解 (Guardian): https://www.theguardian.com/technology/2018/sep/26/uber-hack-fine-driver-data-breach

[A-07] Facebook/Meta FTC $5B制裁 (FTC): https://www.ftc.gov/news-events/news/press-releases/2019/07/ftc-imposes-5-billion-penalty-sweeping-new-privacy-restrictions-facebook

[A-08] Binance $4B+ 解決・CZ有罪 (米司法省): https://www.justice.gov/archives/opa/pr/binance-and-ceo-plead-guilty-federal-charges-4b-resolution

[A-09] DiDi 12億ドル罰金 (Reuters): https://www.reuters.com/technology/china-fines-didi-global-12-bln-violating-data-security-laws-2022-07-21/

[A-10] WeWork Chapter 11 (Guardian): https://www.theguardian.com/business/2023/nov/06/wework-bankruptcy-debt-remote-work

[A-11] Mt. Gox Karpelès有罪 (Reuters): https://www.reuters.com/article/business/japan-court-hands-mt-gox-founder-two-and-a-half-years-suspended-sentence-kyod-idUSKCN1QW08C/

外国人高度人材の日本企業適応支援

内部検討資料

外国人高度人材の日本企業適応支援

— 「根回し」「暗黙の理解」問題への教育的・AI的アプローチ —

清風情報工科学院

2025年2月

作成：平岡憲人

1. エグゼクティブサマリー

先日開催されたインド高度人材パネルディスカッション（日本在住10年以上のパネリスト複数名）において、日本企業における「根回し」と「暗黙の理解」が、長期在住者にとっても依然として最大の壁であるとの報告がなされた。

これは単なる日本語能力の問題ではなく、**意思決定アーキテクチャの構造的差異**に起因する問題である。本資料では、この問題の本質的構造を明らかにした上で、本学院が実施可能な**教育的対応**と**AI活用によるアシスト**の具体策を提案する。

2. 問題の本質 — 何が起きているのか

2-1. 意思決定アーキテクチャの構造比較

日本とインドでは、意思決定のプロセス設計そのものが異なっている。これは文化的な「好み」の問題ではなく、組織運営の基本設計（OS）の違いである。

比較軸	日本型	インド型
意思決定の場	会議前の非公式合意（根回し）	会議の場での論理的議論
会議の機能	事前合意の承認・確認	議論と決定の場
説得の方法	個別の事前調整＋関係性配慮	論理的優位性の提示
情報の伝達	言語化されない文脈（空気）に依存	明示的・論理的に言語化
反対の表明	曖昧表現で間接的に示唆	直接的に論点を指摘

2-2. なぜ10年在住でも解消しないのか

この問題が語学力向上では解決しない理由は、根回しや暗黙の理解が**日本社会の深層構造に根ざしたシステム**だからである。日本の組織は歴史的に、成文化されたルールよりも、関係者間の暗黙の了解と事前調整によって運営されてきた。これは非合理なのではなく、**摩擦を最小化し、実行段階の抵抗をなくすための合理的なアルゴリズム**として機能している。

インドの高度人材は論理的思考力と議論力に秀でているが、「論理的に正しければ場で主張してよい」という前提が日本では通用しない場面がある。問題は論理力の不足ではなく、**その論理力を発揮すべき場所とタイミング**の違いにある。

2-3. 本学院にとっての意味

本学院は日本語教育課程およびIT職業教育課程を通じて、外国人人材の日本企業への就職・定着を支援している。しかし、現行カリキュラムでは日本語能力とIT技術の習得が中心であり、**日本企業の意思決定構造への適応スキル**は体系的に教育されていない。

パネルディスカッションの知見は、就職後の定着率・パフォーマンス向上のために、在学中から「日本的意思決定プロセスの構造理解と実践スキル」を教育すべきことを示している。これは本学院の**教育的付加価値の差別化要因**となりうる。

3. 教育的対応 — 在学中に何を教えるか

以下の施策は、外国人人材の強み（論理的思考力・議論力）を殺さず、日本的意思決定プロセスへの適応力を追加するものである。

3-1. 根回しの構造的理解教育

基本方針

根回しを「日本の不思議な文化」としてではなく、**プロジェクトマネジメントにおけるリスクマネジメント手法**として教える。IT系学生に対しては、「分散処理によるバッチ処理前のデバッグ」というメタファーが有効である。

根回しの3フェーズモデル

フェーズ	目的	具体的行動
第1段階：利害関係者の特定	決裁に影響する人物の洗い出し	ステークホルダーマップの作成
第2段階：事前合意の形成	反対要素の先取りと個別調整	1on1ヒアリング・個別説明
第3段階：表現の調整	反発を生まない提案文の作成	合意済み内容としての再構成

演習案

- 学習者が「論理的に完璧な提案書」を作成する
- 次に同じ提案について「根回し計画書」（誰に・いつ・何を・どの順番で）を作成する
- ロールプレイで反対者役の日本人教員を相手に根回し会話を実践する
- 会議シミュレーションを行い、根回しの有無で結果がどう変わるかを体験する

3-2. 曖昧表現の構造的解読訓練

日本語の曖昧表現を単語の暗記ではなく、**文脈変数を含む推論問題**として教える。これはインド人材の論理的思考力を直接活かせるアプローチである。

曖昧表現の解読フレームワーク

分析軸	観察ポイント	判断への影響
発言者の立場	決裁権の有無・関係性	権限者の曖昧発言はより重い意味を持つ
事前の根回し状況	合意済みかどうか	未根回しでの曖昧発言は警告信号
非言語情報	声のトーン・視線・沈黙の長さ	言葉と非言語の不一致が真意を示す
代替案の提示有無	建設的な対案があるか	対案なき曖昧表現は実質的拒否

具体的な解読例

表面的発言	文脈A（脈あり）	文脈B（実質拒否）
「前向きに検討します」	具体的な次回日程の提案あり	目線が合わず、メモのみ
「ちょっと難しいですね」	代替案を提示しながら	沈黙が続き話題が変わる
「持ち帰ります」	「いつまでに回答する」と明言	「またこちらから連絡します」のみ

3-3. 議論力の「配置転換」訓練

インド人材の議論力・説得力を抑制するのではなく、**発揮する場所を変える**指導を行う。日本では「会議の場で勝つ」より「事前の1on1で勝つ」ことが重要である。

- 同じ提案を3種類の相手（上司・同僚・慎重派）に個別最適化して説明する訓練
- 「論破」ではなく「相手が自分の意見として賛同したくなる説明」の技術を教える
- 相手のメンツを立てながら実質的な合意を取る会話テクニックの実践

3-4. 意思決定フローの可視化教育

日本企業とインド企業の意思決定フローの違いを図式化し、構造的に理解させる。「非公式合意 → 会議 → 形式承認」という日本型フローと、「議論 → 説得 → 決定」というインド型フローの対比を視覚的に示す。構造理解があるだけで、日常業務における摩擦は大幅に減少する。

4. AI的アシスト ― 就職後の実践をどう支援するか

教育だけでは実践現場での対応に限界がある。AIツールによって、外国人人材の日常的な業務コミュニケーションを継続的にサポートする仕組みを構築する。

4-1. 曖昧表現の真意翻訳AI

日本語の曖昧な発言やメールの文面を入力すると、文脈情報（発言者の立場・状況）を加味して**真意の可能性を確率付きで提示する**ツール。LLMのプロンプト設計で実装可能である。

想定入出力例

入力	出力
発言：「前向きに検討します」状況：初回提案、決裁者から	実質的拒否の可能性 60%保留（情報不足）30%本気の検討 10%→ 推奨行動：別ルートからの情報収集
メール：「一旦持ち帰らせていただきます」状況：取引先からの返信	追加プッシュは控える2週間後に別件でコンタクト推奨

4-2. 根回しナビゲーターAI

プロジェクトや提案内容を入力すると、組織情報をもとに**根回しの推奨順序と各人への説明ポイント**を自動提示するツール。

- ステークホルダーの特定と優先順位付け
- 各人向けの説明スクリプト生成（データ重視型・関係性重視型等のパーソナライズ）
- 過去の類似案件の成功パターンの提示

4-3. 提案文の「和風変換」AI

論理的でストレートな文章を、日本的なニュアンスを加えた表現に変換するツール。角を取り、合意済み風にし、相手に逃げ道を残す表現への自動変換を行う。

変換例

変換前（ストレート）	変換後（日本式）
これが最も効率的な解決策であり、即座に実装すべきです	現状を踏まえますと、本案は一つの有力な選択肢と考えられます。皆様のご意見を伺いながら、進め方を検討できればと存じます
この方法は間違っています	この点については、別の視点からも検討する余地があるかもしれません

4-4. 根回しシミュレーターAI（教育×AI統合）

LLMに日本人管理職のペルソナ（メンツを重視する、リスク回避志向、論破されることを嫌う等）を組み込み、学習者がAIを相手に根回しのロールプレイを行うシステム。最大の特長は、**日本人が直接言いにくいフィードバック**をAIが客観的に提供できる点にある。

フィードバック例

「あなたの提案は論理的には完璧でしたが、私の部署のこれまでの努力を否定するような言い回しがあったため、感情的に承認しにくくなりました。次回は、現状の取り組みを評価した上で改善提案として位置づけてみてください。」

5. 実施計画案

5-1. 段階的導入スケジュール

段階	時期	内容	対象
第1段階	2025年度後期	既存カリキュラムへの組込み（曖昧表現解読・意思決定フロー可視化）	日本語上級クラス
第2段階	2026年度前期	根回しロールプレイ演習の導入根回しシミュレーターAIのプロトタイプ開発	就職活動準備クラス
第3段階	2026年度後期	曖昧表現翻訳AIの実装卒業生向けサポートツール提供開始	在校生＋卒業生
第4段階	2027年度以降	企業向け研修プログラムとしての外販法人向けAIツールのライセンス提供	外部企業

5-2. 必要リソース

項目	内容
教材開発	根回し3フェーズモデルのテキスト・ケーススタディ集の作成
AI開発	LLMベースの真意翻訳AI・シミュレーターAIのプロンプト設計と実装
人材	日本企業経験豊富な日本人教員（ロールプレイ相手）の確保
外部連携	就職先企業からの実例・フィードバックの収集体制

6. 期待される効果と差別化

6-1. 学生にとっての価値

- 就職面接において「日本企業の意思決定構造を理解している」ことをアピールできる
- 入社後の適応期間が短縮され、早期に実力を発揮できる
- 長期的な定着率の向上（10年在住でも壁に感じる問題を、事前に構造理解しておける）

6-2. 学校としての差別化

- 日本語＋IT技術に加え、「日本企業適応スキル」を体系的に教える**唯一の教育機関**として差別化
- AIツールの開発・提供により、卒業後も継続的な関係を維持できるビジネスモデル
- 企業向け研修プログラムの外販による新規収益源の創出
- 外国人材の定着支援に取り組む企業への法人営業の切り口

6-3. 社会的意義

日本が外国人高度人材の受入れを拡大する中で、語学力や技術力だけでは定着は実現しない。**意思決定構造の差異**という最も根深い問題に正面から取り組む教育・AIプログラムは、社会的ニーズに合致しており、本学院のブランド価値向上にも寄与する。

7. まとめ

インド高度人材パネルディスカッションの知見は、日本企業における外国人人材の最大の壁が、語学力ではなく**意思決定プロセスの構造的差異**にあることを示している。この問題に対して、本学院は以下の2軸で対応可能である。

アプローチ	内容
教育的対応（在学中）	根回しを構造的に教える、曖昧表現の解読訓練、議論力の配置転換、意思決定フローの可視化
AI的アシスト（就職後）	真意翻訳AI、根回しナビゲーター、和風変換AI、根回しシミュレーター

重要なのは、日本的な曖昧さを「非合理なもの」として押し付けるのではなく、**摩擦最小化のための合理的なアルゴリズム**として構造的に理解させることである。外国人人材の強み（論理力・議論力）を消す必要はない。**発揮する場所を会議前に移すだけ**である。

本プログラムの段階的導入により、本学院は「外国人人材が日本企業で真に活躍するための実践的スキルを教える教育機関」として、独自のポジションを確立できる。

以上