WEEK 8 ／ 事前学習 最終回

プロジェクトマネジメント

テーマ あらゆる領域の判断に根拠を持たせ、プロジェクトを前に進め続ける

目的① マネジメントの判断も、技術判断と同じく根拠で決める軸を持つ

目的② 依存をほどく道具を持ち帰り、直後の作業期間の計画づくりで使う

講義は今日で終わり、直後からチームの作業期間に入る

WEEK 6

宇宙開発とシステムズエンジニアリング

→

WEEK 7 ｜ 前回

ソフトウェアエンジニアリング

→

WEEK 8 ｜ 今日・講義最終回

プロジェクトマネジメント

依存を描く PERT 図とマージン。
コードを書く前のインターフェース定義

→

以降

作業期間・チーム開発、そして合宿へ

1

プロジェクトマネジメントとは何か

標準の定義を読み替えると判断の元手に行き着き、その元手が有限であることから
トレードオフが生まれ、その正体は依存関係だと分かる。

PMIによるプロジェクトマネジメントの定義

プロジェクトの要求事項を満たすために、知識・スキル・ツール・技法を
プロジェクト活動へ適用すること。

PMI（Project Management Institute）によるプロジェクトマネジメントの定義。PMI は標準体系 PMBOK ガイド（Project Management Body of Knowledge、現行第8版）をまとめた団体

出典：PMI, “What Is Project Management?”

つまりどういうこと？

1.2 定義の読み替え

マネジメントとは、
あらゆる領域の判断に根拠を持たせ、前に進め続けることである

標準の定義（PMBOK ガイド 第8版）

プロジェクトの**要求事項を満たすために**、知識・スキル・ツール・技法を**プロジェクト活動に適用すること**。

本ゼミの言葉への読み替え

要求事項を満たすために	⟶	要求から根拠をつなげる
プロジェクト活動に	⟶	あらゆる領域に
適用すること	⟶	適用するときに根拠を持つ

↓

あらゆる領域の判断に根拠を持たせ、プロジェクトを前に進め続けること

判断の元手、プロジェクト資源（人・時間・予算）は後から増やせない

では、その判断は何に対して行うのか。

人

4人

チーム編成は固定

時間

4泊5日＋事前学習

合宿の日程は固定

予算

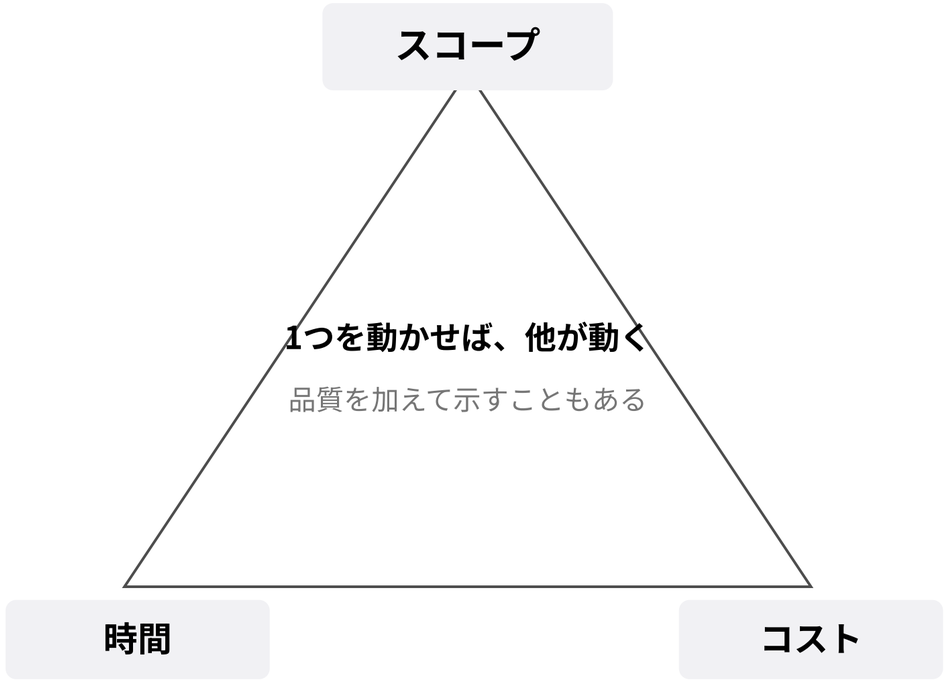
機材は支給

機材は運営が用意し、追加購入はない

判断＝有限の資源をどこに配るか（配る先の判断 ＋ 配り方の判断）

1.4 トレードオフ

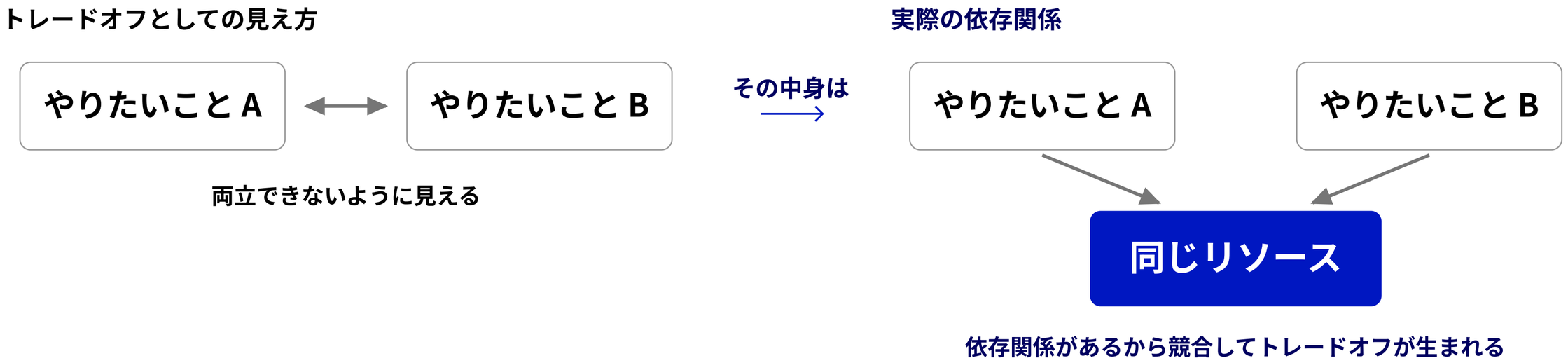
同じ元手を取り合うから、スコープ・時間・コストは同時には立てられない



- **トレードオフは、資源が有限であることの帰結。**
  同じ人の時間、同じ日程、同じ機材という有限の元手を取り合うから生じる
- **代表例が三重制約（スコープ・時間・コスト）。**
  1つを動かせば他が動く。品質を加えて4軸で示すこともある

製造業では **QCD**（品質・コスト・納期）という考え方がよく使われる。

トレードオフは見え方で、実際の構造は依存関係



トレードオフを判断するために、まずその仕組となる依存関係を知る。
依存関係を解きほぐし、まず最適解を考え、どうしても両立できないものはトレードオフを考える。

まず依存を解き、次にリスクに備え、それでも足りないときに選ぶ

①

依存関係を解きほぐす

待ちをなくし、同じ人数と日数で進む量を増やす

2章

→

②

リスクに備える

見積もりの外れと前提の崩れから、手戻りと停止を減らす

3章

→

③

最後に選ぶ

何を諾めて何を守るか、結局目的は何か、その基準はサクセスクライテリア。

Week 6

①②で最適解を探し、それでも両立しないものだけを③で判断する

動かせるのはスコープだけ —— その軸は Week 6 の3段階で刻んである

コスト＝主に人／4人で固定　機材は運営が支給　時間＝合宿の日程で固定　⟶ **動かせるのはスコープだけ**

EXTRA
追従・連続運用
移動する目標への追従など、フルの上の発展目標。定義は各チーム

FULL
制御＋撮影＋画像送信
姿勢を制御して対象を撮影し、画像を送る（要求の達成）

MINIMUM
姿勢推定＋テレメトリ
地上から状態が見える＝全運用の最低条件

- **上位の段は、下位を包含する。**
  フルはミニマムを含む。地上から状態が見えることが、あらゆる運用の最低条件（Week 6）
- **削る順序も、この入れ子が持つ。**
  遅れたら外側の段から諦め、範囲を内側へ絞る（デスコープ。型は Week 6 のまま）

7つのドメインは並行して走り続け、本回はその中を順番に通っていく

PMBOK 第8版の7つのパフォーマンス・ドメイン ／ 並行して走り続ける

横＝並行して管理し続けるドメイン、重ねた①②③＝本回で決めていく順番。独自の分類は立てず、標準の区分の上に通り道を描いている

1.9 この章のまとめ

ここは覚えて次へ：判断・有限の資源・トレードオフの正体

① **実際の活動＝判断に根拠を持たせ、前に進め続けること。**元手のプロジェクト資源（人・時間・予算）は有限で、判断は配る先＋配り方の両方。

② **トレードオフの正体は、依存の構造にある。**ほぐせば消える対立と、外から固定されて残る本物の制約を見分ける。

③ **削るのは最後の手段。**まず依存を解き、次にリスクに備え、それでも足りないときに削る。

2

依存

今日の中心。まず依存を読み、そのうえで2つの手でほどく。

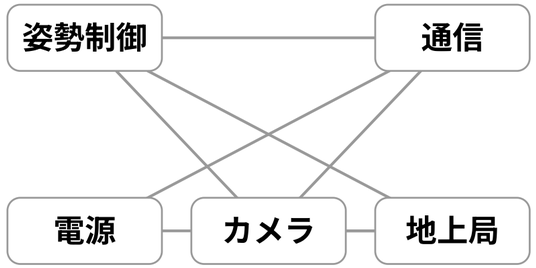
前半 依存を読む

線で描き、最長経路を読み、何からやるかを決める

後半 依存をほどく2つの手

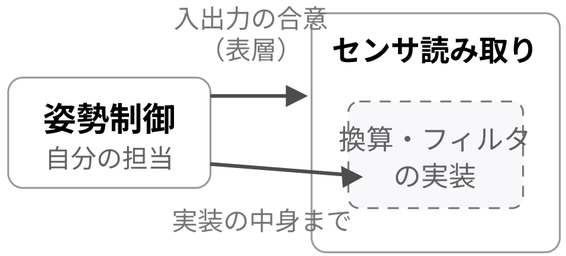
内部事情を切り離す・輪を切る

密結合の難しさは、数・内部事情・堂々巡りの3つが重なっている



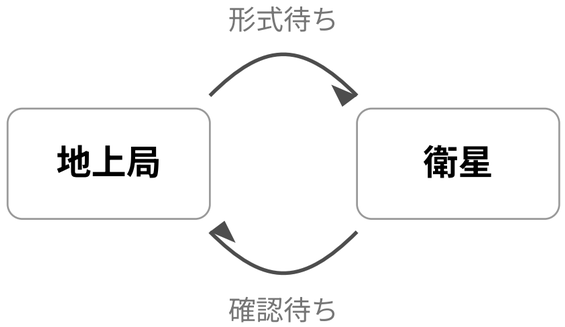
①絡む相手が多い。

サブシステムの数だけ依存の相手がいる



②内部事情まで絡む。

例：姿勢角を返す関数の入出力で済むはずが、換算・フィルタの実装まで待ってしまう

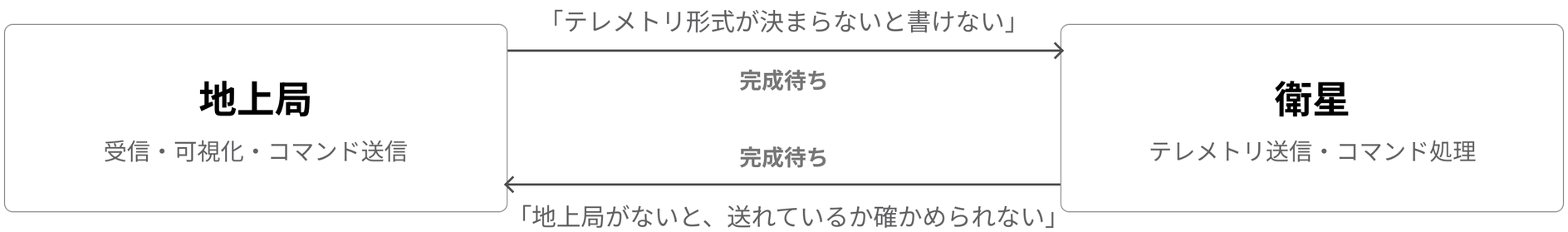


③依存が堂々巡りする。

互いに相手の完成を待つ循環ができ、両方が止まる

2.2 堂々巡り（循環依存）

地上局と衛星は、互いに相手の完成を待って両方止まる



互いの完成を待ち、両方が止まる。誰も怠けていないのに、何も進まない。

待ちは、人的資源を消費しながら何も生まない

待っている間も、Cさんの時間は消費され続ける。作業の失敗と違い、やり直しても何も残らない

- **待ちは有限の資源を最も大きく捨てる。**
  作業の失敗はやり直せば成果が残るが、待ちは時間だけが消える
- **4人では、1人の詰まりが全体を止める。**
  小さいチームほど、依存の1本が計画全体の首を絞める

マネジメントの本質は、タスクの依存関係をほどくことにある

同じ作業量でも、順番だけで完了が45分変わる

カレーライスとサラダを作る。工程はどちらも同じで、着手の順番だけを変える。

段取り①

米を研いで炊飯を始め、待つ間にカレーとサラダを作る

60分で完成

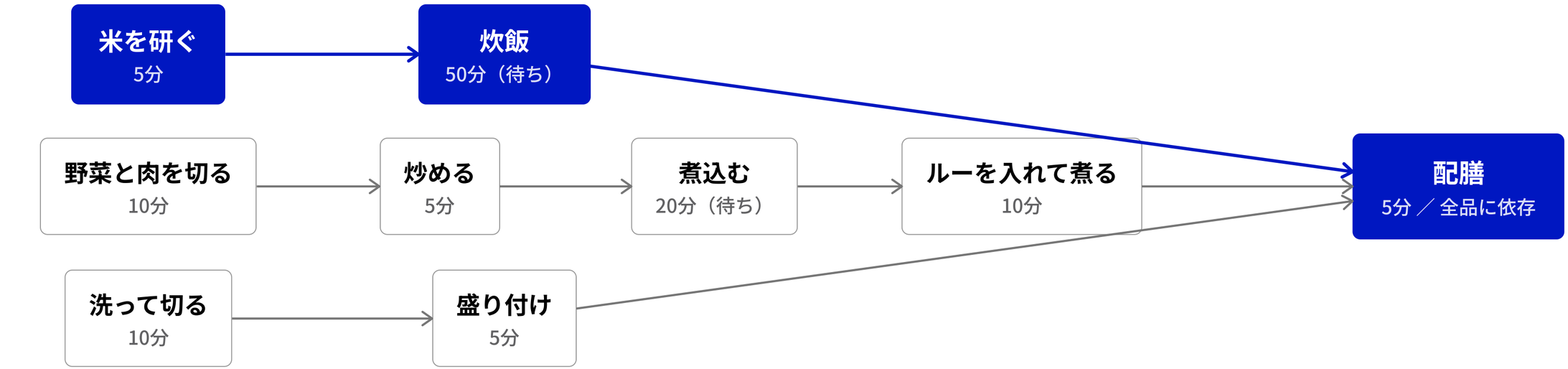
段取り②

カレーを作り終えてから、米を研ぐ

105分で完成

作業量は1分も変わらない。着手の順番だけが、完了時期を決めている

クリティカルパスは、手数最多のカレーではなく炊飯を含むご飯の鎖



クリティカルパス：ご飯の鎖 55分 ＋ 配膳 5分 ＝ 60分　　余裕あり：カレー 45分、サラダ 15分（約40分の余裕）

炊飯・煮込みは人手を拘束しない待ち。そこへサラダを差し込める

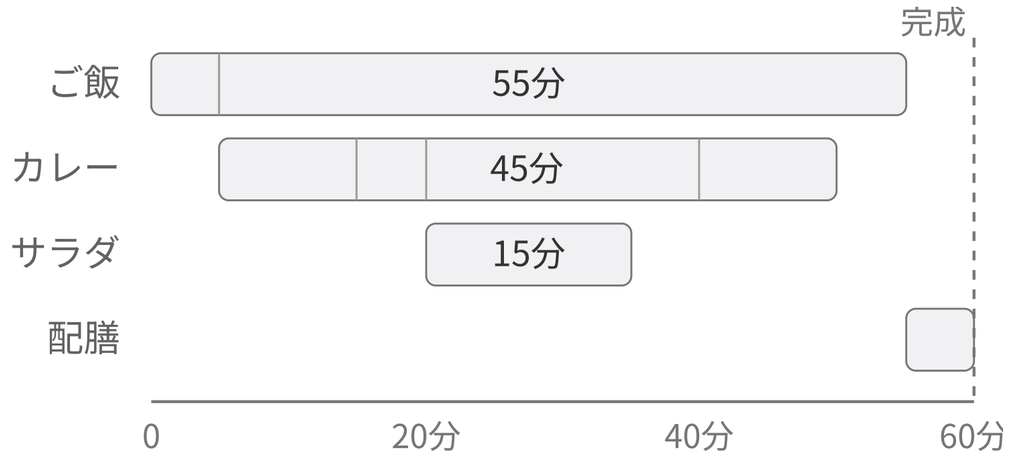
PERT 図
依存関係を矢印で置いた図。実務ではこの呼び名で通じる

クリティカルパス
線をたどった最長経路。合計時間が全体の完了時期を決める

余裕
経路の外なら遅れても、全体は遅れない幅

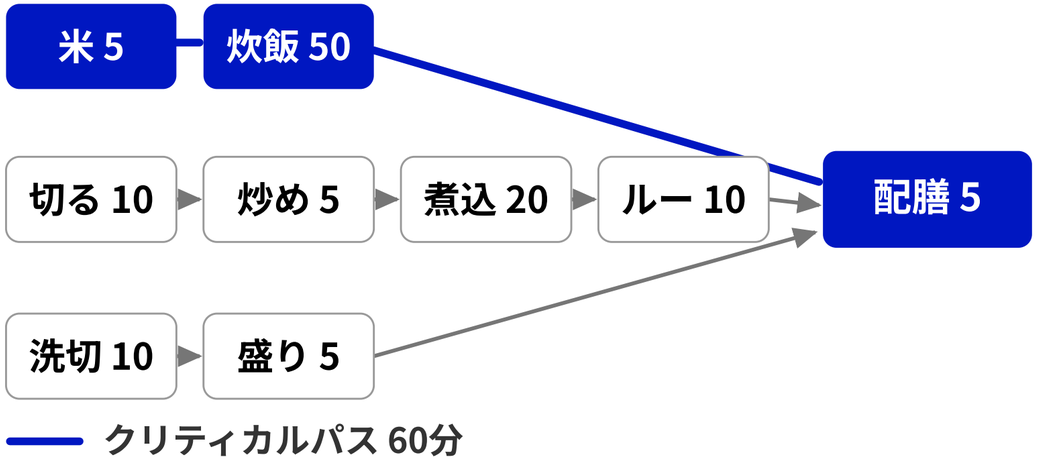
同じ食卓の計画でも、ガントチャートには依存の線が引かれない

ガントチャートで描く



いつ何をするかと進捗は見える。矢印がないので、炊飯が完了時期を決めていることは読めない。

PERT 図で描く



依存の線と最長経路が見える。時間軸は持たないので、着手の時刻は別に決める。

全タスクがテレコマの完成に依存する自分たちの計画では、依存を描けない図は計画の最大の敵を映せない。

標準体系の呼び名は PDM と CPM、三点見積もりを載せた派生が本来の PERT

PDM	プレシデンス・ダイアグラム法。依存関係を四角と矢印で描く図法。実務の「PERT 図」の多くはこれ
CPM	クリティカルパス法。依存図から最長経路を求める手法
本来の PERT	Program Evaluation and Review Technique。依存図に三点見積もり（楽観・最頻・悲観）を載せた派生（3章で触れる）

ふだんの呼び名は**「PERT 図」**でよい。呼び名の違いで詰まる必要はない。

PDM で描いた依存図
実務の「PERT 図」の多くはこれ

＋

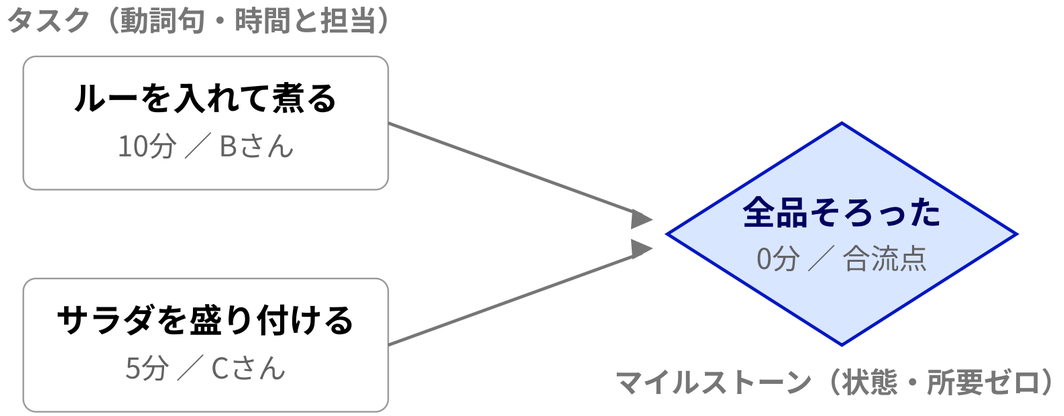
三点見積もり
楽観・最頻・悲観（3章で触れる）

＝

本来の PERT

ノードは作業で統一し、合流点だけをマイルストーンにする

- **タスクノードは動詞句で書く。**
  所要時間と担当をノードに載せる。粒度は「1人が半日」。矢印は「前が終わらないと始められない」だけを表す
- **マイルストーンは合流点だけ。**
  状態の名詞句、所要時間ゼロ。「カレー完成」「配膳の直前」のような数個のゲートに閉じ込める
- **2つの流儀を混ぜない。**
  ノードを作業、矢印を順序にする流儀が AON（Activity on Node。2.7 の PDM がこの方式で、本デッキも AON）。矢印を作業、ノードを状態にする流儀が AOA（Activity on Arrow）。1枚の図の中で混在させない



迷ったときの判定　時間と担当が置ける → **タスク**　／　「〜が動いている」という条件文になる → **マイルストーン**

各タスクに完了条件（何が観測できたら終わりか）を書けば、その条件がそのまま状態になる

スケジュール管理の道具は他にもある ― 使いどころが違う

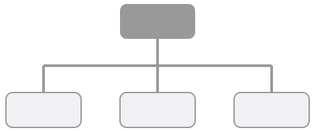
ガントチャート

タスクを時間軸上のバーで並べる図

○ 大人数・長期の進捗共有に強い
△ 依存関係が主役ではない。少人数・短期間では得るものが少ない

WBS

Work Breakdown Structure。成果物・作業を階層的に分解して網羅する図



○ 抜け漏れの点検に強く、PERT 図のノードを書き出す下ごしらえになる
△ 一度で分解し切るのは難しい。粗く始めて更新し続ける

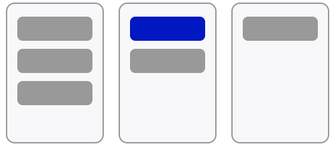
マイルストーンチャート

節目（マイルストーン）だけを時間軸に打つ図

○ 長期日程が一目で共有できる
△ 途中の作業量や依存は見えない。少人数・短期間では得るものが少ない

カンバン

「未着手・作業中・完了」の列でタスク札を動かす板



○ いま誰が何をしているかが見える
△ 完了時期の見通しは示さない

2.10 何からやるか

数十のタスクを4人で回すとき、どれから手を付けるか？

難しそうなものから

重いものを先に片づけておく

得意なものから

手が動くところから進める

締め切りが近いものから

期日の順に並べて消化する

作業量は同じでも、順番で完了時期は変わる。では、何を根拠に順番を決めるか。

2.11 優先度の3つの型

「何からやるか」は、クリティカルパス・リスクの高さ・依存の根元で決める

①

クリティカルパス上を優先する

経路上の1日の遅れは、全体の1日の遅れになる。経路の外には余裕がある

②

リスクの高いものを先に潰す

外れが早く分かれば、作り直す時間が残っている。終盤の発覚には打つ手がない

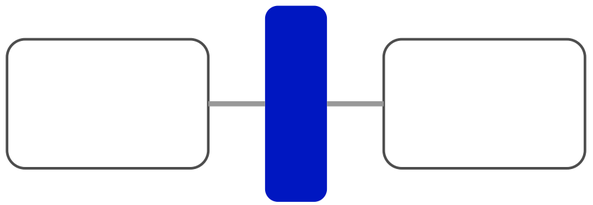
③

依存の根元を先に固める

根元が動くと、下流の作業がまとめて作り直しになる。テレコマの形式がこの位置

3つとも、依存の構造とリスクから順番の根拠を取っている。難しさや得意さは、順番の根拠にならない。

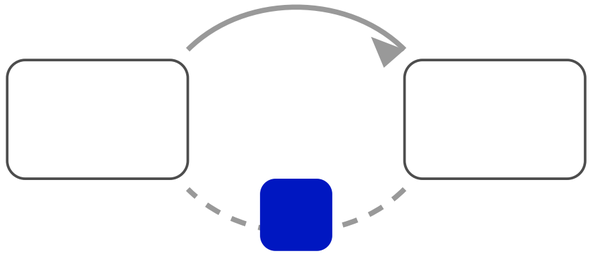
読めた依存を、2つの手でほどく



手①

内部事情を切り離す

インターフェースを先に決め、境界で作業を分割する



手②

輪を切る

堂々巡りの1つを仮に決め打ちして、輪を切る

コードを書く前にインターフェースを定義し、境界で作業を分割する

衛星側の実装
内側は自由に作り、差し替えられる

→

合意したインターフェース
依存を、内部事情から表層の合意へ縮める

←

地上局側の実装
合意だけを見て、独立に進む

パケットフォーマットの合意に最低限含めるもの（行の中身は例。何を載せるかはチームの設計判断）

項目名	型とサイズ	単位	送信周期
attitude_angle	int16（2 byte）	0.01 deg	1 Hz
…（項目ごとに1行ずつ、チームで合意する）			

Week 7 の「呼ぶ側を変えずに中身を差し替えられる」を、コードの層からチームの層へ持ち上げた操作。契約まで固めることは、要求と制約を人にもAIにも渡せる形にする作業でもある。

- **一度決めて終わりではない。**
  合意は必ず変わる。定義の正本をどこに置き、変えたときにどう周知して切り替えるかを、あらかじめチームで決める（置き場所と手順の中身は各チームの設計判断）

輪の中の1つを仮に決め打ちすれば、両側が同時に動き出す

堂々巡りのまま

形式が決まらない

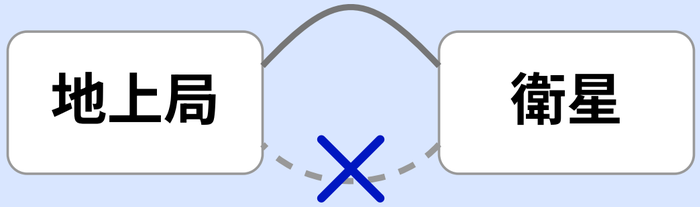
地上局 ⇄ 衛星

届いているか確かめられない

互いに相手が決まるのを待って、両方が止まる

いったん決め打ち →

輪が切れる



形式を仮に決め打ち

仮に決めた形式を前提に、両側が同時に動き出す

決め打ちだと明示する

決めた中身と一緒に「これは仮」であることを共有する

決め直す時期を決める

いつ、何が分かった時点で見直すかを、決め打ちと同時に置く

外しても戻しやすい側を選ぶ

崩れたときに影響が小さいところで切る

決め打ちを動かす道具が、ダミーデータとモック（Week 7）。実物ができたら差し替える。あわせて、依存の向きは一方向に揃え、最小の経路を先に1本通してから太らせる。

2.15 Conway の法則

担当を分けた線が、そのままシステムの分かれ目になる

上：チームの分担　下：できあがるシステム構造

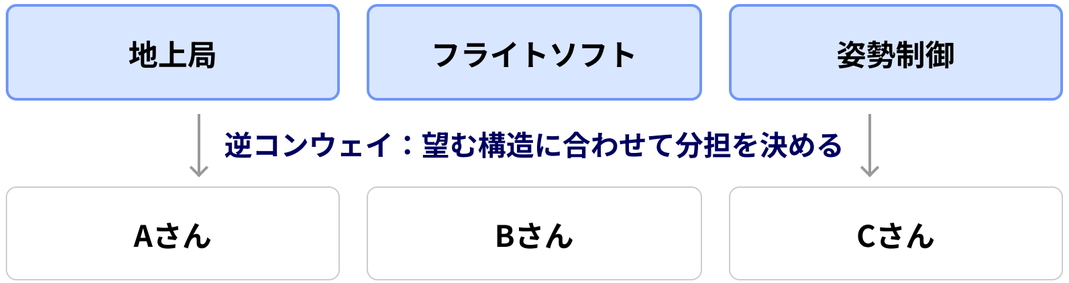
分担の境界 ＝ モジュールの境界 ＝ インターフェース

システムを設計する組織は、その組織のコミュニケーション構造を写した構造の設計を生む（Conway, 1968）。

2.16 逆コンウェイ

誰に割り当てるかと、どこで依存を切るかは、1つの判断である

- **逆コンウェイ：分担を先に設計する。**
  望むアーキテクチャに合わせて、先に担当の分け方を決める
- **手①と同じ操作である。**
  インターフェースを先に決めて境界で分割することと、分担を決めることは、同じ1つの判断の両面
- **分担の見直しは、設計変更でもある。**
  担当を組み替えるときは、切り直されるインターフェースも一緒に見直す



上：望むシステム構造　下：そこから導く分担（2.15 の逆向き）

2.17 この章のまとめ

ここは覚えて次へ：依存を読む道具と、ほどく2つの手

① **待ちが資源を最も大きく捨てる。**マネジメントの本質はタスクの依存関係をほどくこと。

② **クリティカルパスが完了時期を決める。**「何からやるか」は、経路上・リスクの高さ・依存の根元で決める。

③ **ほどく手は2つ。**切り離し（インターフェース）・輪を切る（決め打ち）。可視化（PERT 図）は、ほどく前に依存を読む道具。

④ **分担の線がシステムの分かれ目になる。**割り当てと、どこで依存を切るかは1つの判断。

3

リスクに備える

依存をほどいても、見積もりは外れる。手戻りと停止に先に手を打ち、残量を見張る。

不確実性のうち名指しできるものがリスク、名指しできない残りには使い道を決めない資源で備える

不確実性（上位の語）

リスク＝名指しできるもの

洗い出して評価し、対応を作れる事象

残り＝名指しできない外れ → 使い道を決めない資源で備える

- **名指しできたリスクには、対応を作る。**
  避ける・減らす・移す・受け入れるの4つから選ぶ
- **名指しできない残りには、使い道を決めない資源を残す。**
  洗い出しは尽きず、名指しできたリスクの見積もりそのものも外れる

リスクは、洗い出して、起こりやすさと影響で評価して、対応を決める

段階1

洗い出す

何が起きたら計画が止まるかを挙げる

→

段階2

評価する

起こりやすさと影響の大きさで見る

→

段階3

対応を決める

4つの型から選ぶ（次のスライド）

Week 6 で技術的な壊れ方（故障モード）に当てたこの形を、**計画・スケジュール・資源**へ当てる。例は、電池が想定より持たない、部品が壊れる、メンバーが1日離脱する。

3.3 対応の4つの型

対応の型は、避ける・減らす・移す・受け入れるの4つ

① **避ける**

そのやり方を採らない。間に合わない見込みの部品を使わない設計に変える

② **減らす**

起こりやすさか影響を小さくする。最小の経路を先に1本通しておく

③ **移す**

負う先を変える。実務では保険や外注、合宿では運営が用意する予備機材

④ **受け入れる**

起きたら対処すると決め、備えだけ置く。マージンがこれに当たる

備えないという判断も、根拠を言えるなら判断である。起こりやすさも影響も小さいものに資源を割くほうが、配り方としては下手になる。

3.4 マージン

プロジェクト資源の一部は、使い道を決めずに残す —— これがマージン

割り当て済みの資源（使い道が決まっている）	マージン

使い道が決まっていないことが機能。どんな外れ方にも充てられる

スケジュール予備

見積もりの外れに備えて、予定を詰め切らず時間を残す

リソースバジェットの余裕

電力・通信・メモリの収支に余裕を明示的に載せる。ぴったりの収支は少しの外れで破綻する

マージンは怠慢ではなく、有限の資源の配分に関する意思決定である。

3.5 参考：三点見積もり

楽観・最頻・悲観の3点で、見積もりを幅として扱う

楽観値	最頻値	悲観値
15分	**20**分	**30**分
うまくいけば	たいてい	火の通りが悪ければ

カレーの煮込み（2.5 の例）を1点の20分ではなく、15〜30分の幅で扱う。この3点を PERT 図に載せた派生が、本来の PERT（2.7）。

計画の窓は、マージンの残量とクリティカルパスの進み具合

運用の窓（Week 6）

運用中、衛星の中は覗けない。テレメトリだけが観測の窓。

計画の窓（今日）

計画のズレは、マージンの残量とクリティカルパス上の進み具合に現れる。

マージンを使い切って遅れたら、何を削るかは**デスコープの階段（Week 6）**が持っている。残量の観測は、その判断を起動する材料。

3.7 スナップショット

計画を更新する前に、前の版をスナップショットとして残す

当初計画
作業期間で描いた PERT 図

→

更新の前に記録
ホワイトボードなら、更新前に写真を1枚撮るだけでよい

→

更新後の計画
合宿では必ず更新される

- **当初計画と実績の差が、Day 5 の素材になる。**
  どこが外れ、そのときどう決め直したか。差分が残っていなければ、時間の使い方の意図を語れない

3.8 決め直しの輪

残量を見て決め直したら、また依存とリスクの取りこぼしへ戻る

① **取りこぼしを潰す**
依存をほどく（2章）、リスクに手を打つ（3章前半）

→

② **使い道を決めない資源を残す**
マージンを置く（3.4）

→

③ **残量と進み具合を見張る**
マージンの残量と、クリティカルパス上の進み（3.6）

→

④ **ズレたら決め直す**
足りなければスコープを削る（デスコープの型は Week 6）

↑ 1.6「削る前に問うこと」へ戻る

この輪が回り続けることが、1.2 で置いた定義「あらゆる領域の判断に根拠を持たせ、プロジェクトを前に進め続けること」の中身である。

直前の輪をどれだけ粗く回すかを決めることが、計画の立て方そのものである

予測型

全体を先に計画し、その計画を基準に進める。全体像が比較的よく見えている場合に向く。

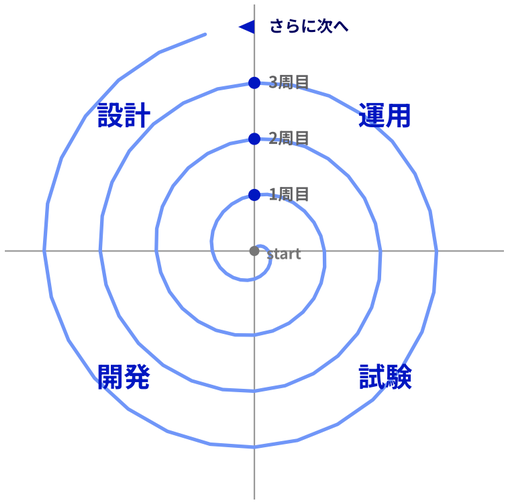
適応型（アジャイル）

粗く計画し、短い反復の中で学びながら頻繁に決め直す。やってみて初めて分かることが多い場合に向く。

ハイブリッド

対象や部分によって、予測型と適応型を組み合わせる。

どれを採るかは、全体像がどれだけ見えているかで決まる**設計判断**。本ゼミでは適応型を中心に、固定された日程や合流点には予測型の考え方も使う。どの部分を固定し、どの部分を反復するかも設計判断で、ダブルダイヤモンドを繰り返す形（4.2）と同じ向き。



適応型の一周（Week 6 の図の再掲）：設計→開発→試験→運用を小さく一周し、周ごとに育てる

3.10 この章のまとめ

ここは覚えて次へ：名指しできるリスクへの手と、名指しできない残りへの備え

① **名指しできるリスクは、洗い出して評価して対応を決める。**対応の型は、避ける・減らす・移す・受け入れるの4つ。

② **名指しできない残りには、使い道を決めない資源を残す。**これがマージン。使い道が決まっていないことが機能。

③ **マージンは、残量を見張って初めて機能する。**計画の窓はマージンの残量とクリティカルパス上の進み具合。更新前の計画はスナップショットとして残す。

④ **決め直したら、また取りこぼしを潰す段へ戻る。**この輪が回り続けることが「前に進め続ける」の中身。

まとめ

今日の核：判断に根拠を持たせ、順番を守って、前に進め続ける

①

判断に根拠を持たせ、前に進め続ける。元手のプロジェクト資源（人・時間・予算）は有限で、判断は配る先＋配り方。

②

トレードオフの正体は、依存の構造にある。ほぐせば消える対立と、外から固定されて残る本物の制約を見分ける。

③

削る前に、取りこぼしを潰す。依存は PERT 図で読み、インターフェースとダミーの2つの手でほどく。

④

リスクは名指しして対応を作り、残りはマージンで備える。残量を見張り、更新前の計画はスナップショットとして残して決め直す。決め直したら取りこぼしへ戻り、この輪が回り続けることが「前に進め続ける」の中身。

宿題

Week 6 からの通し課題を、「どう進めるか」の観点で考え直す

通し課題（Week 6 から）

模擬衛星の初回打ち上げに必要な機能を考える

→

第3稿：今日が足す観点

お題は Week 6 と同じで、**今日はどう進めるか（依存と計画）**

Week 6　SE｜初回に必要な機能は何か →

Week 7　SWE｜どう作り、どこで検証するか →

Week 8　PM｜どう進めるか（依存と計画）

何からやるか

依存とクリティカルパスから、作る順序を決める

どこで依存を切るか

インターフェースを先に決める位置を選ぶ

どこにマージンを置くか

外れる前提で、置き場所を決める

完了条件：作る順序・依存を切る位置・マージンの置き場所を、それぞれ何を根拠に決めたかを言えること。

実装は課さない。提出は作業期間の最初のチーム作業の前で、各自の案を持ち寄る。所要は60分を目安に

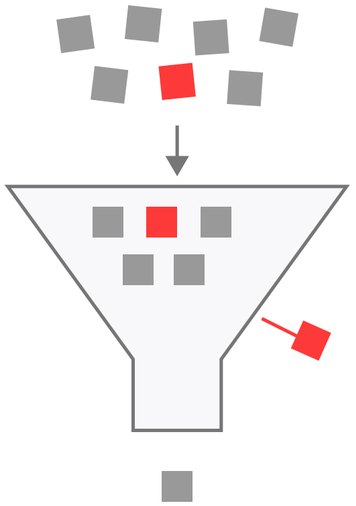
4

参考：共通講義のまとめ

宿題や今後の作業期間で必要なスタンス

AIとの協働は、検問ではなく共同のエンジニアリングである

✕ 検問



AIの出力（高速・大量）

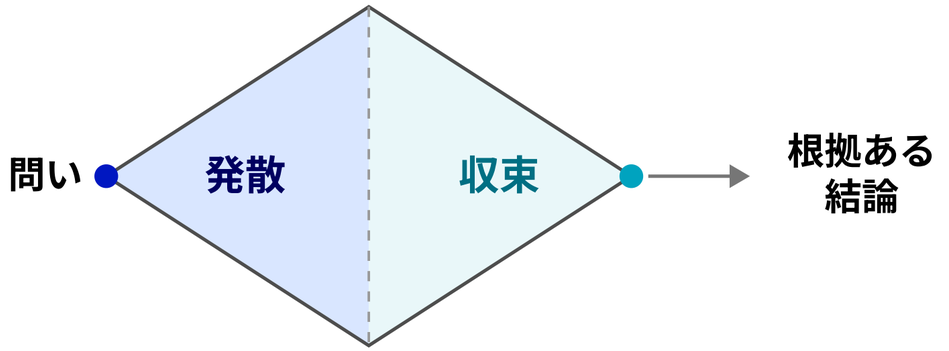
人間が1件ずつ審査
量で破綻する

見逃し

「毎回チェック」は量で破綻し、「全部信じる」は判断の明け渡し。
審査の根拠は設計されていない。

◯ 共同のエンジニアリング

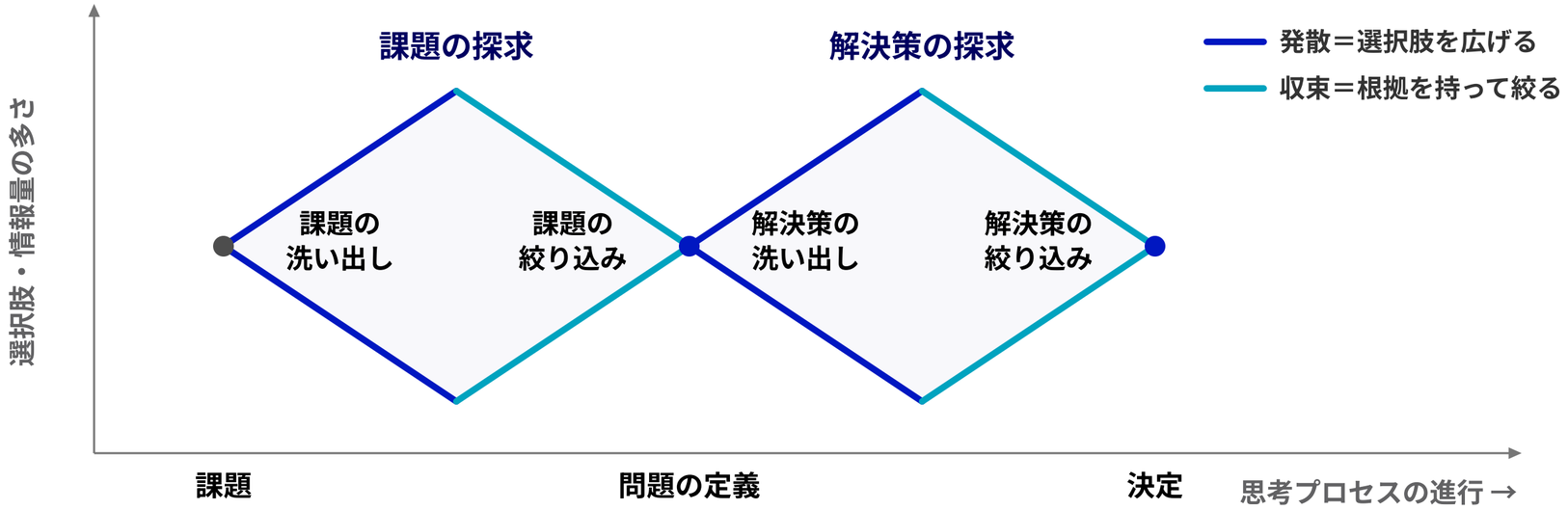
AI＋人間で回す



広げるのも、確かめるのも、AIに任せてよい

人間が要求と契約（テスト・インターフェース）を固め、内側を委任する。人間の役割は「広げたか、何で確かめたか」を説明できること。

開発は、発散と収束を2回繰り返す形で進む



この形を1周で終えず繰り返す。作業期間の計画づくりが、このループの設計をAIと対等に分担する最初の実践になる。

付録 出典

参考文献と図版の出所

- Project Management Institute（PMI）『PMBOK（Project Management Body of Knowledge）ガイド』第8版 —— プロジェクトマネジメントの定義、パフォーマンス・ドメイン、三重制約、PDM・CPM・三点見積もりの用語
- Melvin E. Conway, “How Do Committees Invent?”, Datamation, 1968 —— Conway の法則（2.15）
- Matthew Skelton, Manuel Pais, “Team Topologies”, IT Revolution Press, 2019 —— チームタイプとインタラクションモード（2.15 の台本で名前のみ紹介）
- Design Council, “Framework for Innovation”（ダブルダイヤモンド、4.2）—— 本ゼミでの位置づけの正典はゼミ設計資料 concepts.md
- 図版はすべて本資料内で作図（デジタル庁ダッシュボードデザイン実践ガイドブック準拠のデザインシステムを使用）。写真・ネット取得画像は使用していない