衛星開発ゼミ

WEEK 6

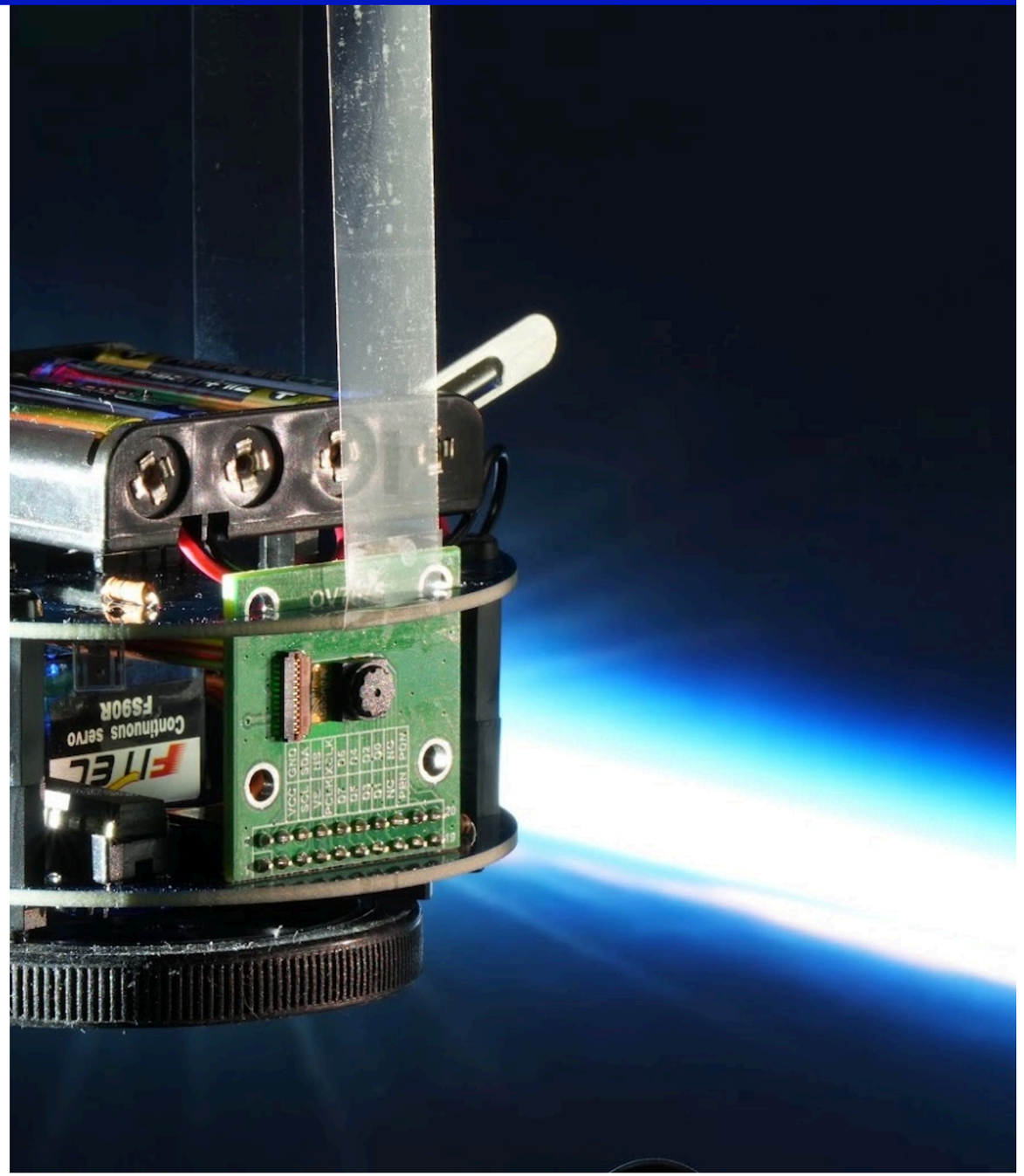
宇宙開発とシステムズエンジニアリング

模擬衛星のミッション要求と宇宙機の制約を理解し、その開発プロセスを導く

テーマ 要求と制約から、開発プロセスが決まる

目的① 模擬衛星と本物の宇宙開発のつながりをつかむ

目的② 宇宙機を根拠をもって開発するプロセスを学ぶ



0.1 前回までの復習

第1段階：要素技術という手札をそろえた

WEEK 1	**マイコン**	数値の読み書きという基本原理を知り、原理に基づいて設計の手札を広げる姿勢を学んだ
WEEK 2	**ハードウェア間通信**	UART・SPI・I2C など、機器どうしをつなぐ通信の型を学んだ
WEEK 3	**センサとの通信とデバッグ**	センサと実際に通信し、原理の理解に基づくデバッグの姿勢を学んだ
WEEK 4	**リアルタイムシステム**	リアルタイムシステムと状態遷移の型を知り、巨人の肩に立つ姿勢を学んだ
WEEK 5	**Wi-Fi通信とネットワークプロトコル**	離れた相手とデータをやりとりする仕組みと、その約束事を学んだ

↓

第1段階：要素技術と根拠のある設計

0.2 今回の位置づけ

第2段階：根拠のあるエンジニアリング

第2段階：根拠のあるエンジニアリング

WEEK 6 ／ 今日

進め方と観測の設計

宇宙機の要求と制約に基づく開発プロセスと、観測の重要さまで

WEEK 7

検証の仕組み化

テストとシミュレーションで、誤りを人間より先に仕組みが見つける状態を作る

第3段階

WEEK 8

計画とマネジメント

スケジュールとマージンを設計し、チームで開発を進める

1

やりたいことを実現する

ミッションは決まっている。そこから、どう作るか？

やりたいこと

対象を撮影して、画像を手に入れる

動くもの

実装された模擬衛星システム

衛星は複数の系に分かれ、模擬衛星もこの区分を引き継ぐ

色付きの4系（ADCS・COMM・C&DH・Payload）を、ゼミのソフトウェア開発で扱う

- **ADCS** ＝ Attitude Determination and Control System
  姿勢を決定（推定）し、制御する系
- **C&DH** ＝ Command and Data Handling
  コマンド処理とデータ取り扱いを担う系
- **COMM** ＝ Communication
  地上局との通信を担う系

参考

復習：事前に見てもらった動画

https://www.youtube.com/watch?v=6KcV1C1Ui5s

ミッション部 Payload
今回はカメラ

姿勢系 ADCS
姿勢を推定し制御

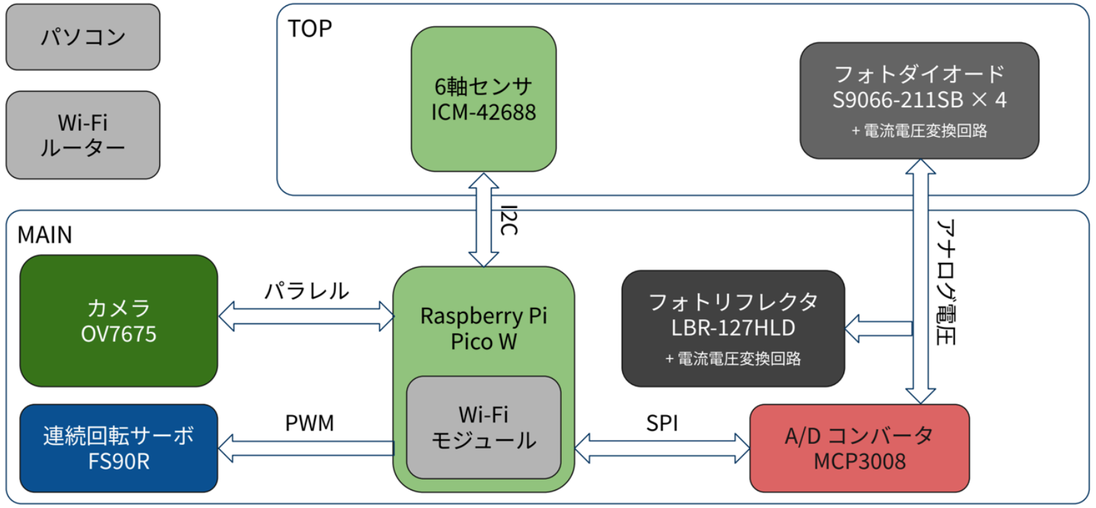
データ処理系 C&DH
中核の計算機 OBC が全体を統括

通信系 COMM
地上局とテレメトリ・コマンド

電源系
太陽電池＋充電池／生命線

構体・熱制御系
今回は主題から外す

実衛星の構成を、模擬衛星にあてはめて考える



サブシステムが1枚の基板にぶら下がる（TOP／MAIN 基板）

- **C&DH：Pico W（RP2040）**
  全体を統括する計算機（OBC）
- **ADCS：ジャイロ＋太陽センサ＋サーボ**
  ICM-42688（I2C）／フォトダイオード×4（ADC）／サーボ＋タイヤ（PWM）
- **COMM：マイコン内蔵 Wi-Fi**
  PC とテレメトリ・コマンドを送受信
- **Payload：カメラ**
  OV7675 （パラレル＋I2C）

システム＝要素の集合＋要素間の関係

要素の集合（寄せ集め）

- センサ
- サーボ
- カメラ
- 無線

部品単体では動作する。
でもバラバラのままでは、ミッションを達成できない。

関係を設計 →

関係を持ったシステム

センサ → OBC → サーボ

カメラ → OBC → 無線

サブシステムをどう組み合わせ、システムとするか。
この組み合わせ方は自明ではない。

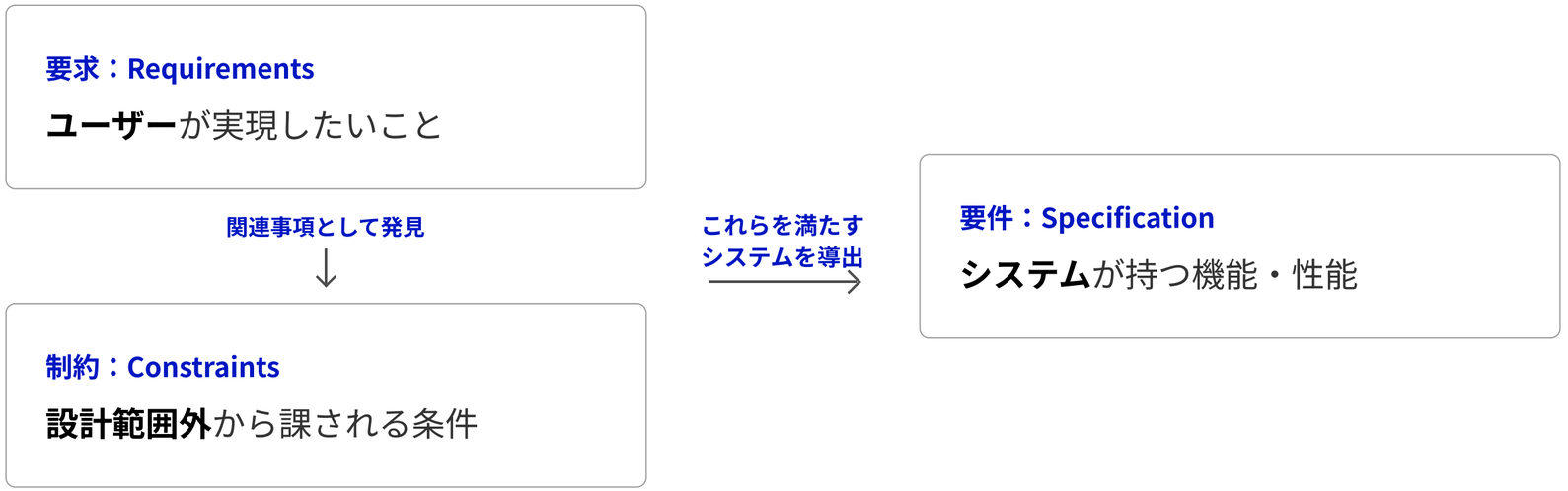
2

「やりたいこと」から出発する

まずはプロジェクトの全ての根拠の出発点を明確にする

要求・制約・要件を分けて整理する

3語は、主語も焦点も整理方法も違う。



発展：本当はミッション要求そのものを考えるプロセスがある。このゼミでは要求はこちらから与える。

太陽との相対位置が既知の天体の画像を取得する

- **要求は「やりたいこと」だけ。**
  どうやって実現するか（姿勢制御方法・通信手段）は書かれていない
- **主語はユーザー。**
  「人間が」画像を見たい。主語は衛星ではない。

「宇宙空間」の中で対象の画像を取得する

成功の定義を、先に段階で決めておく

0 か 100 か

成功
全部できた

失敗
1つでも欠けた

これでは開発にあたって優先順位がつけられない。

サクセスクライテリア（3段）

高 ↓ 低

- **エクストラサクセス：** 余剰リソースを活用して狙う**計画的ボーナス**
- **フルサクセス：** 本来の要求仕様を完遂する**本来のゴール**
- **ミニマムサクセス：** プロジェクトの価値が存在する**絶対死守ライン**

エクストラ：「肥大化（＝無駄な開発）」を防ぐ**天井**
ミニマム：「全損（＝プロジェクト失敗）」を防ぐ**底**
これを事前に合意してプロジェクトを進めることが重要

2.4 3段階の中身

サクセスクライテリアはレベルの高低だけでなく、範囲を絞るイメージ

上位の段は下位を**包含**する（フルはミニマムを、エクストラはフルを含む）ことで、
やる/やらない の範囲を状況に応じて選択できる（**スコープ管理**）。

EXTRA

追従・連続運用

移動する目標への追従、複数パスの安定運用など。定義は各チーム

FULL

制御＋撮影＋画像送信

姿勢を制御して目標を向き、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

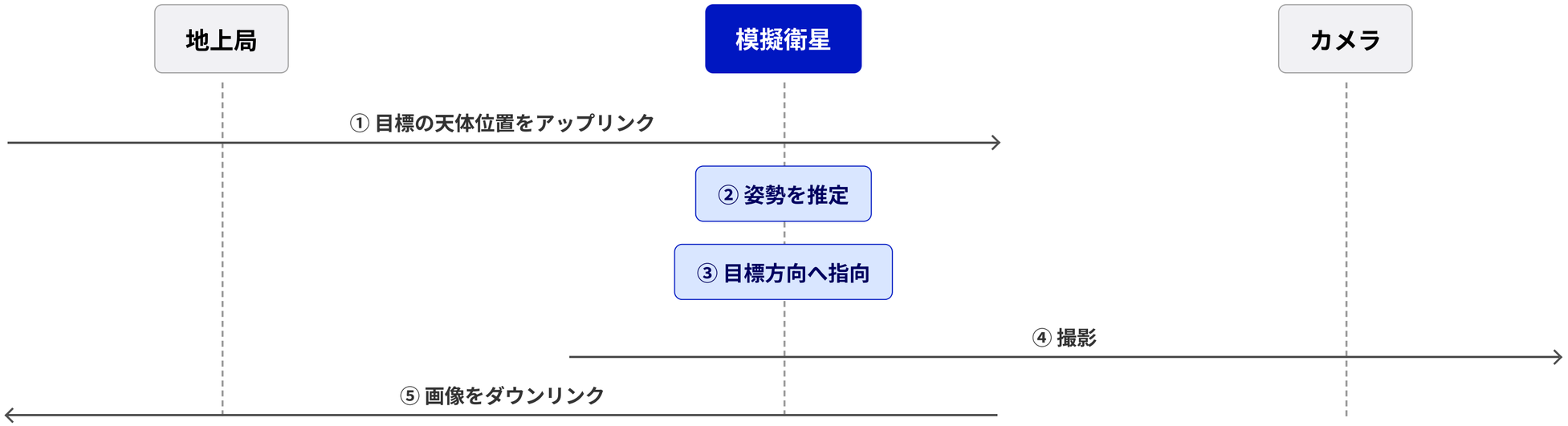
地上から状態が見える＝全運用の最低条件

後で回収 なぜこれが模擬衛星のミニマムサクセスとして適切か？

要求・制約・要件を具体化する

区分	なぜ、どうやって具体化する？	例
要求	ミッションの軸は自明でも、 関連する要求を見落としていないか「考える」	機体の状態を把握する／対象の画像を手に入れる
制約	プロジェクトとシステムの置かれる状況により 決まっているが、自分たちで「発見する」	ハードは設計済み／電波だけ届く
要件	要求(やりたいこと)と制約(置かれた状況)から、 自分たちで「導出する」	どう姿勢を推定・制御し、どんなデータをどう送るか

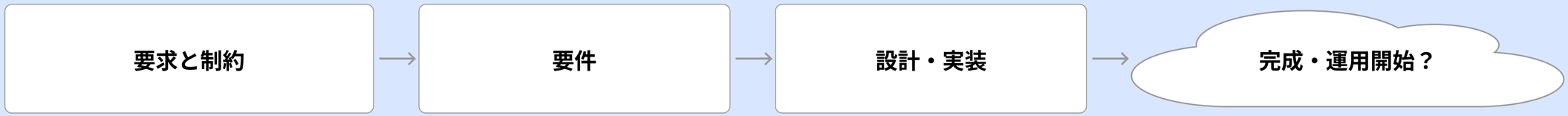
ミッションシーケンスへ具体化すると、要件が見える



各ステップが要件の問いを生む（「静定した」と言える条件は？ 画像はどう分割する？）。完成した要件一覧は、皆さんが洗い出す。

3

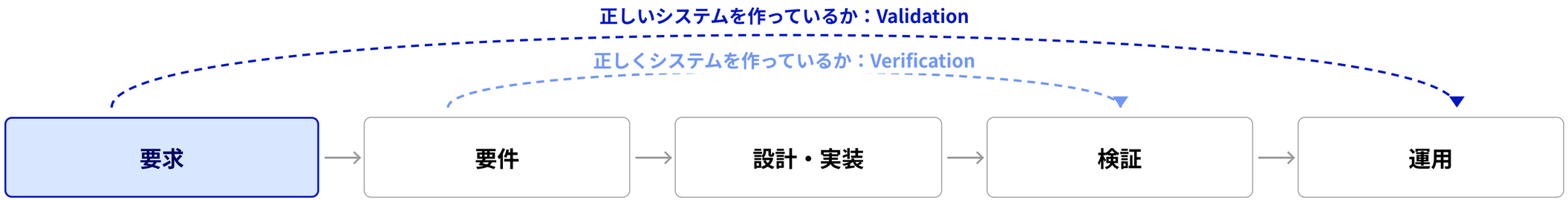
「やりたいこと」を実現するプロセス



「やりたいこと」に基づいて進めば、根拠が地続きのまま運用まで行けるはず。
しかし、一度でも間違った設計・実装をすると**最後に**破綻してしまう。

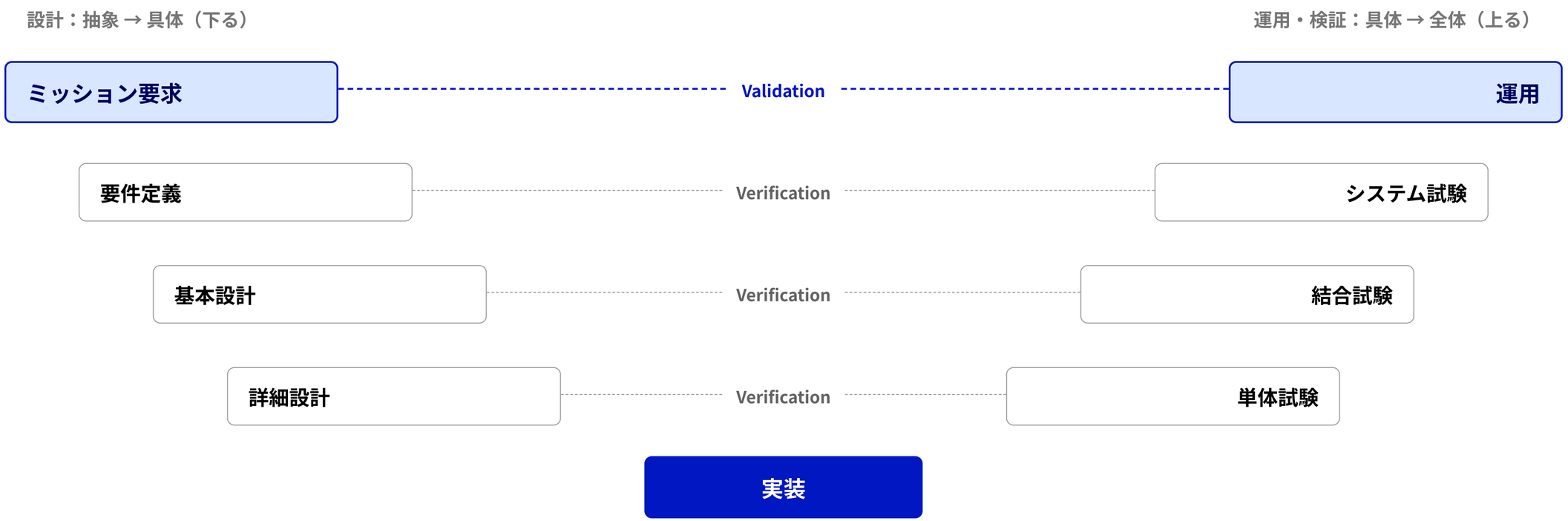
設計・実装したものと、「やりたいこと」との整合を確認する

要求・要件にもとづき設計・実装した後、本当にその通りできているか確認する。
開発のプロセス全てが根拠で繋がっていること（トレーサビリティ）が重要。



- **要求**に合っているか：**正しい**システムを作っているか
  要求＝システムを使ってユーザーがしたいこと
- **要件**に合っているか：**正しく**システムを作っているか
  要件＝システムができること

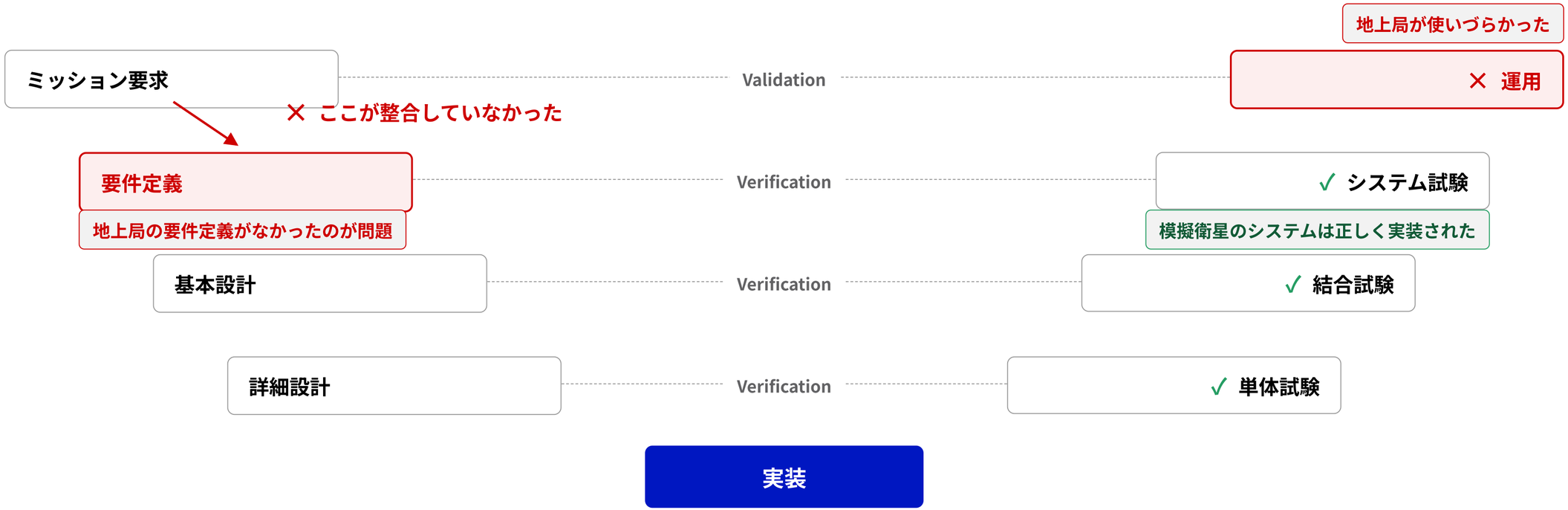
一直線を折り曲げてレイヤーを対応させる



3.3 もし整合しなかったら

後半で問題が見つかるほど、手戻りは大きい

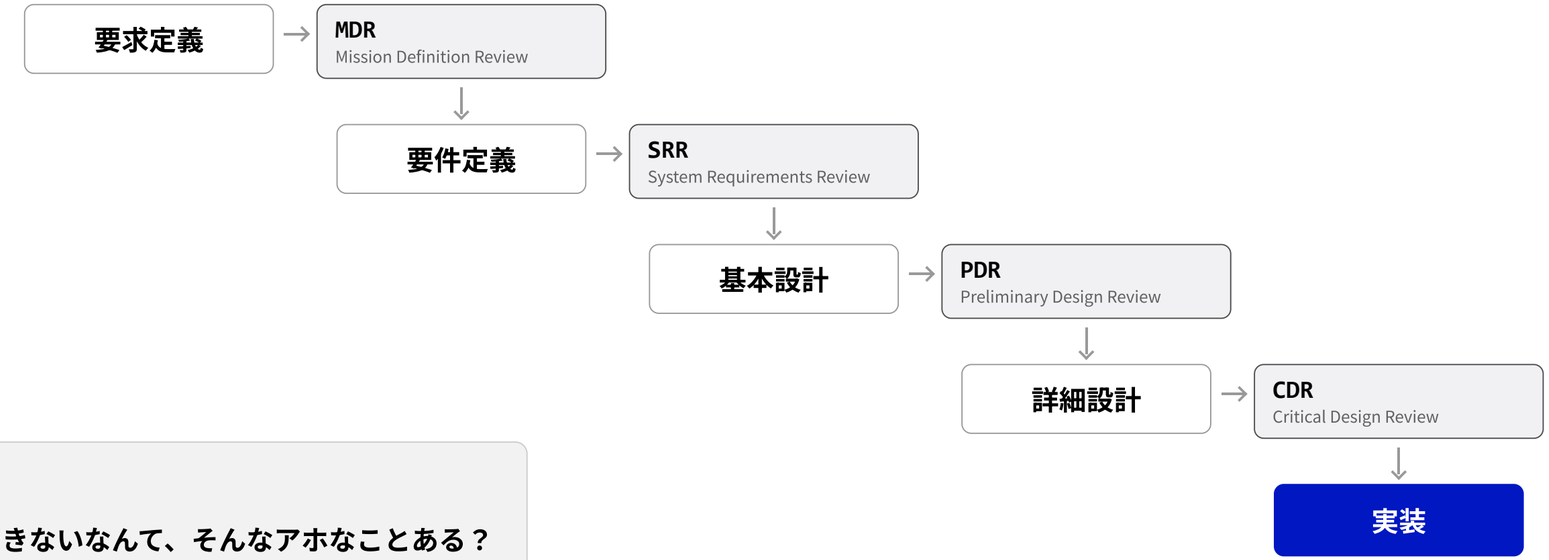
最後の運用で見つかった問題は、最初の要件定義のミスが原因。やり直しが大変。



※ レイヤーごとの対応は、試験↔その根拠であり、試験の**対象**は常に完成した**実装**。

工程の節目ごとに、審査（デザインレビュー）を挟む

- 最初は試験の手段がないから、**考え抜くしかない**。
- 次のフェーズに進んでよいか（**Go / No-Go**）を審査・決定する会を設ける。



発展

作るまで検証できないなんて、そんなアホなことある？

Model-Based Systems Engineering という手法がある。
少なくとも模擬衛星では、作った方が早いので省略。

DRでも、設計・実装を始めてからでも、不整合に気づくことは必ずある

設計は順方向に進むが、変更は逆方向に辿って影響を洗い出してから行う。これを**変更管理**と呼ぶ。
このためには、まず順方向の依存関係が完全に整理されている必要がある。（**トレーサビリティが重要**）
判断だけでなく**「何を根拠にそう決めたか」**が残っていることが重要。

順方向：何もないところから作る

電源への要求：電圧は○○V → **電気設計：電池は単三4本** → **構造設計：単三4本を積む構造**

逆方向：後から変更を加える（変更管理）

電源への要求：電圧は同じで変更可 ← **電気設計：電池を単四に** ← **構造設計：単三4本は載らない**

もし、電圧ではなく**容量**を根拠に単三を選定していたら、この変更はできない。
根拠の書き残しが日頃からなければ、影響範囲を辿れず変更管理は破綻する。

発展

判断と、その根拠を残す記録の型

ADR（Architecture Decision Record、アーキテクチャ決定記録）。判断が覆ったときは上書きせず、新しい記録で置き換え、古い方に「置き換え済み」の印と参照を残す。審査して承認する体系には**変更管理委員会（CCB）**がある。

3.6 対策3：「やってみなくちゃ分からない」ならやってみる

小さく一周して、周ごとに育てる

最初から「考える」のは難しい。

- 経験が浅いと「考え抜く」ことはできない
- 分からない／何が分からないか分からない

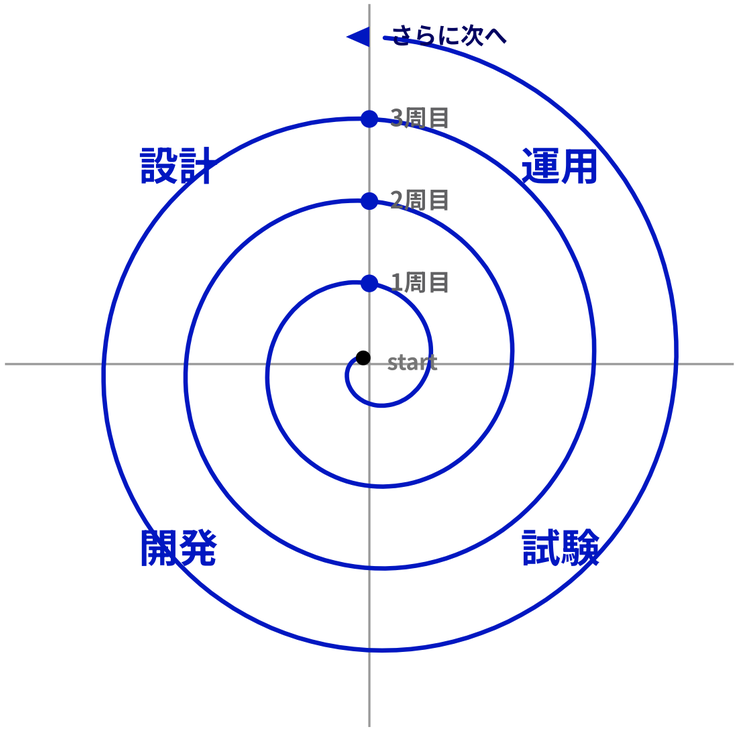
全部を作り切る前に小さく一周して確かめる。

- 設計→開発→試験→運用を**小さく一周**
- 完璧な一発を狙わず、**早く回して育てる**

Week 7 予告

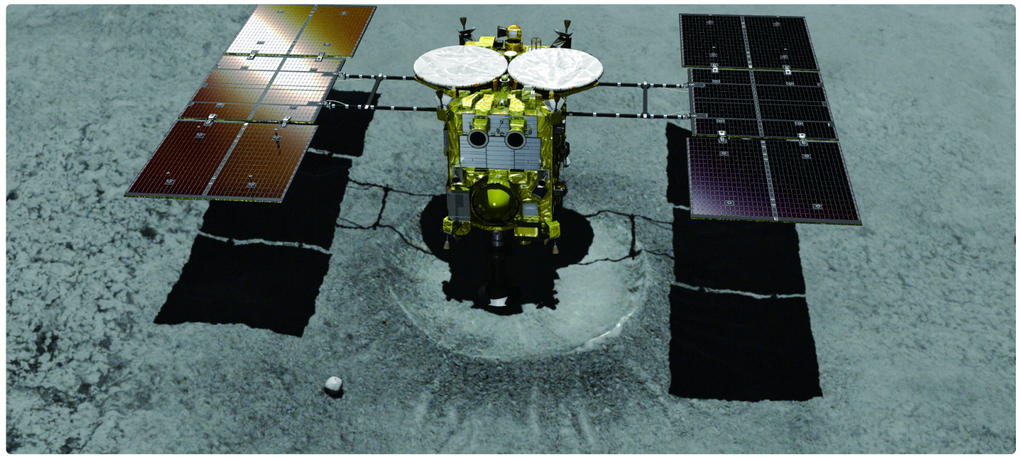
運用前にもやってみる

小さく速い「やってみる」ループを、打ち上げの手前にいくつも回す仕組み



中心から外へ。周を重ねるごとに、機能と完成度が育つ。
一周の中の流れはV字と同じ。

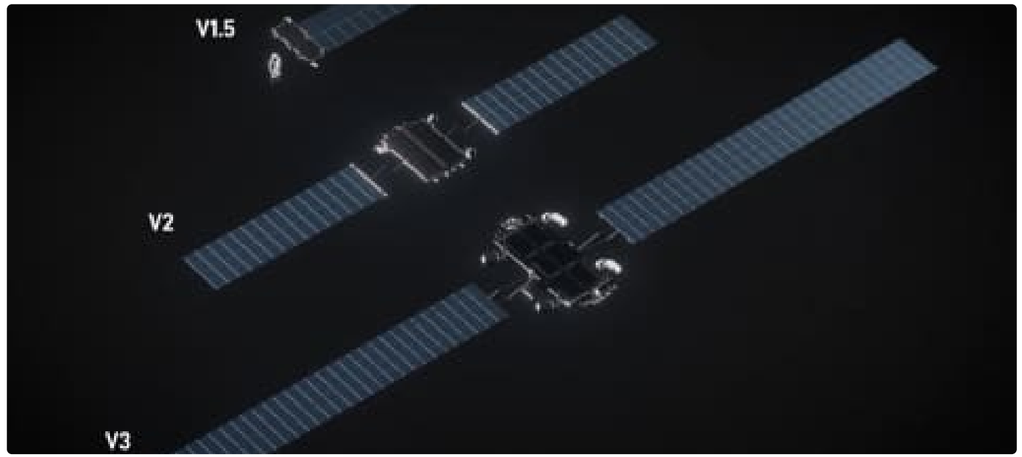
開発プロセスそのものも、設計判断の対象



©JAXA

事前検討型（一点ものの探査機など）

背景：失敗する余裕がない／長期運用で、高速なフィードバックが不可
方針：運用前にシミュレーションと検討を徹底的に行う



©SpaceX

実機反復型（SpaceX・小型衛星ベンチャーなど）

背景：実証の回数を確保できる（コスト・スケジュール）
方針：実機で得た結果を次の反復に活かす

グラデーション：じっくり検討するタイプの はやぶさ にも"2"がある。反復型でも、実証で裏付けられた設計の根拠がある。

4

宇宙って、なんで特別難しいの？

「極限環境」・「複雑系」・「非修理系」

極限環境である：放射線・熱・真空・振動

極限環境であり、それに耐えるには専門的な設計が必要。
放射線試験、熱真空試験、振動試験、それぞれ検証は行うが、非常に手間がかかる。

参考

復習：事前に見てもらった動画

https://www.youtube.com/watch?v=6KcV1C1Ui5s

放射線

宇宙線が半導体を突き抜け、メモリのビット反転や突発リセット

真空・熱サイクル

日向と日陰で百度以上の温度差。真空では熱が逃げにくい

打上げ振動

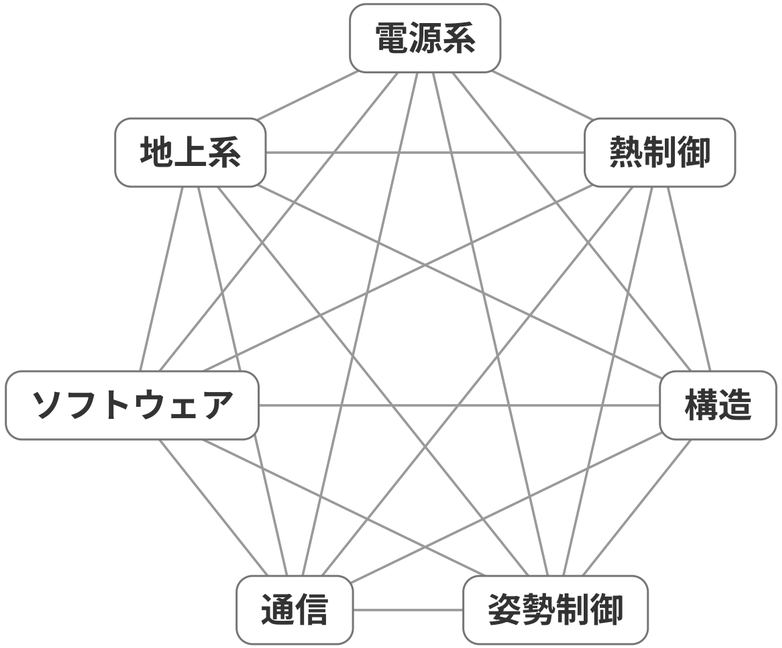
ロケットの激しい加速と振動が、配線や部品を壊しうる

電源

太陽電池と充電池だけ。電力収支が崩れたら終わり

複数の領域が密に絡み合う

- 全てが**密結合**で、簡単には切り分けられない。
- **境界（インターフェース）**を決めて整理するコストが大きい。
- サブシステム単体が正常でも、**統合して初めて出る問題**がある。

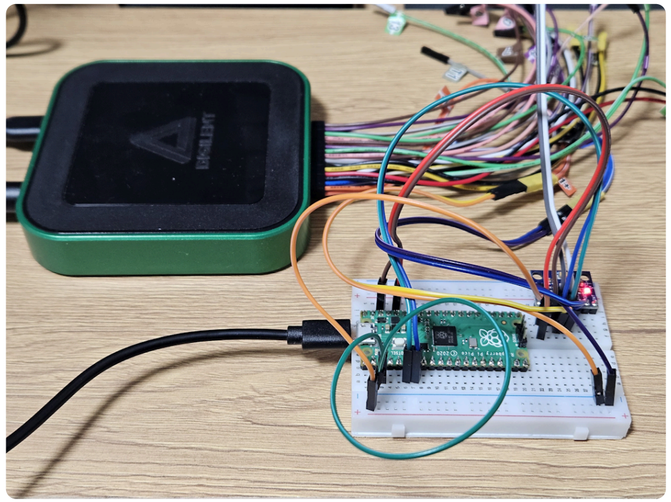


4.3 離れた場所にある

修理できない、運用に入ったら触れない

アクセス（観測・介入）する手段を、後から増やせない。

地上の開発



壊れたら

- **観測**：オシロスコープ・ロジックアナライザ・デバッガなどを導入できる
- **介入**：その場で配線・コードを変えて修正できる

衛星（軌道上）

壊れたら

- **観測**：電波のテレメトリだけ。完全に故障したら何も分からない。
- **介入**：ハードウェアは触れず、電波でコマンドを打つだけ

4.4 難しさの整理

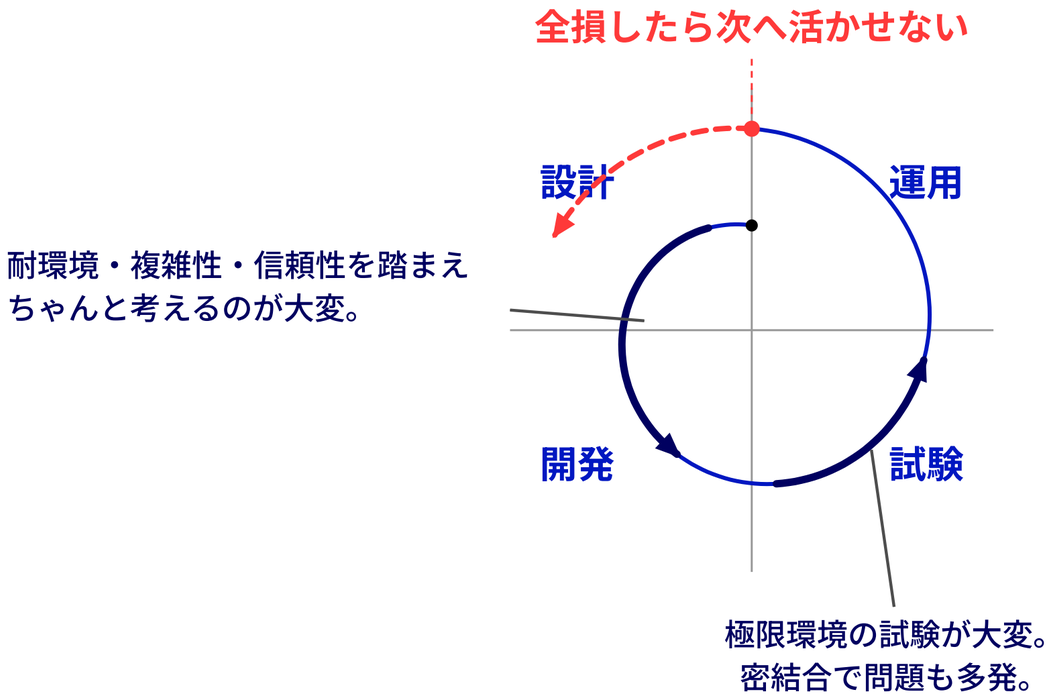
ちゃんと考えるのも難しい、やって試すのも難しい

宇宙開発の特徴が、開発プロセスの各フェーズでの難しさにそれぞれ繋がる。

宇宙の特徴	ちゃんと考える難しさ	やってみて試す難しさ	模擬衛星での再現
極限環境	本質的・専門的な設計が必要	検証に手間がかかる	✕ 環境は地上
複雑なシステム	様々な専門家との調整が必要	全部組み上げないと試験できない	✓ エレキ/メカ/ソフト程度は絡む
非修理系	自律制御がある程度必要	通信がダメなら全損	✓ 運用ルールで再現する

宇宙では「やってみたけど分からなかった」が最悪

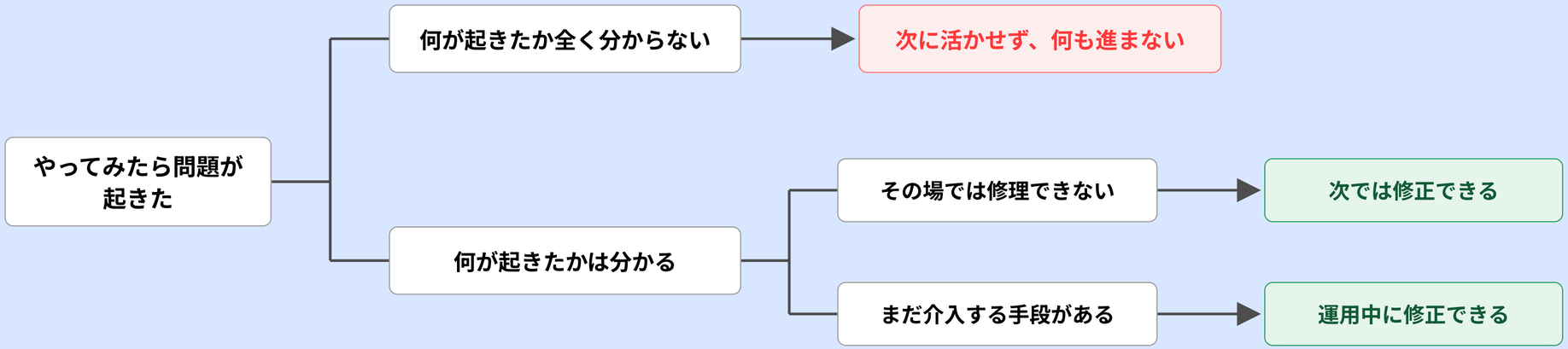
全フェーズに難しさはある。しかし「面倒」ではなく**「詰む」ポイント**がある。



5

「やってみる」ために必要なこと

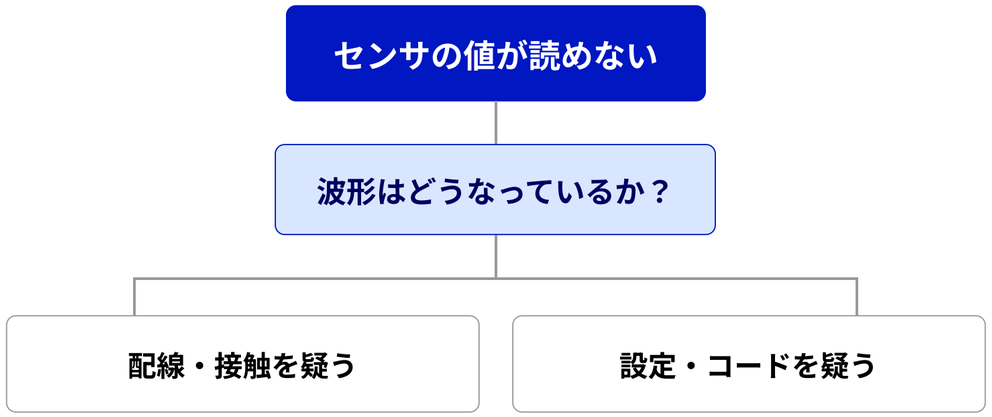
試行の価値を確保する。



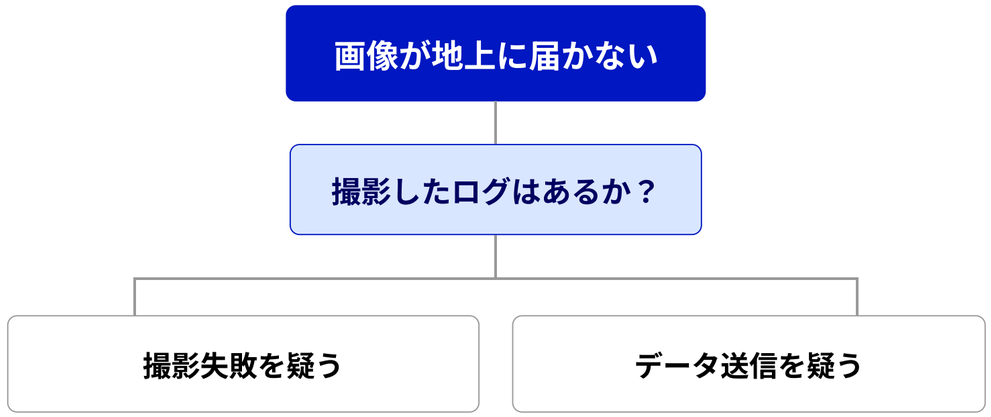
原因を突き止めるプロセスは地上と同じ

「観測した事実」をもとにシステムを切り分ける

地上：センサ通信トラブルの切り分け



衛星：FTA（故障の木解析）



宇宙機で起こったことを観測する手段の設計が重要

宇宙機のテレメトリ・通信の重要性

- 問題発生後に観測手段を追加して確認できない。
- 故障で動作が停止したら、内部を調査できない。

開発中

print デバッグ　波形観測　…

いつでも中を観測できる。観測手段を追加して検証できる。

運用中

テレメトリ（唯一の観測手段）

中を覗けない。だから観測点を事前に設計する。

ユーザーが欲しいデータと、そのためのシステムに必要なデータ

ミッションデータ

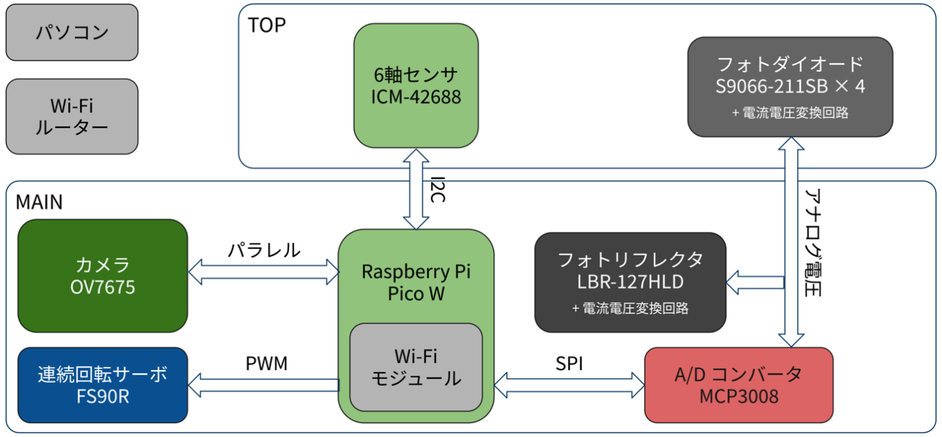
システムのユーザーが欲しいもの

結局、写真が撮れるのならそれだけで良い

HK（ハウスキーピング）テレメトリ

衛星の状態を知るためのデータ（運用・切り分けに使う）

電源・姿勢など機体自身の状態



2系統の記録を、両方とも地上へ送る

5.4 データを回収するためには

多少問題が起こっても、データは回収する。衛星を生存させる。

データを取って反復するなら当然必要。難しいミッションを一発で成功させる場合も、まずは衛星を生存させるために必要。
これが全ての土台になる。

どう直し、どう介入するか

OTA　コマンド体系

観測と検知の深掘り

HK の項目・頻度　リミット

壊れる前に洗い出す

FMEA　リスク管理

宇宙特有のトラブルから復旧する

冗長系　ウォッチドッグ　セーフモード

今日は項目だけ、詳細は Week 6・7

6

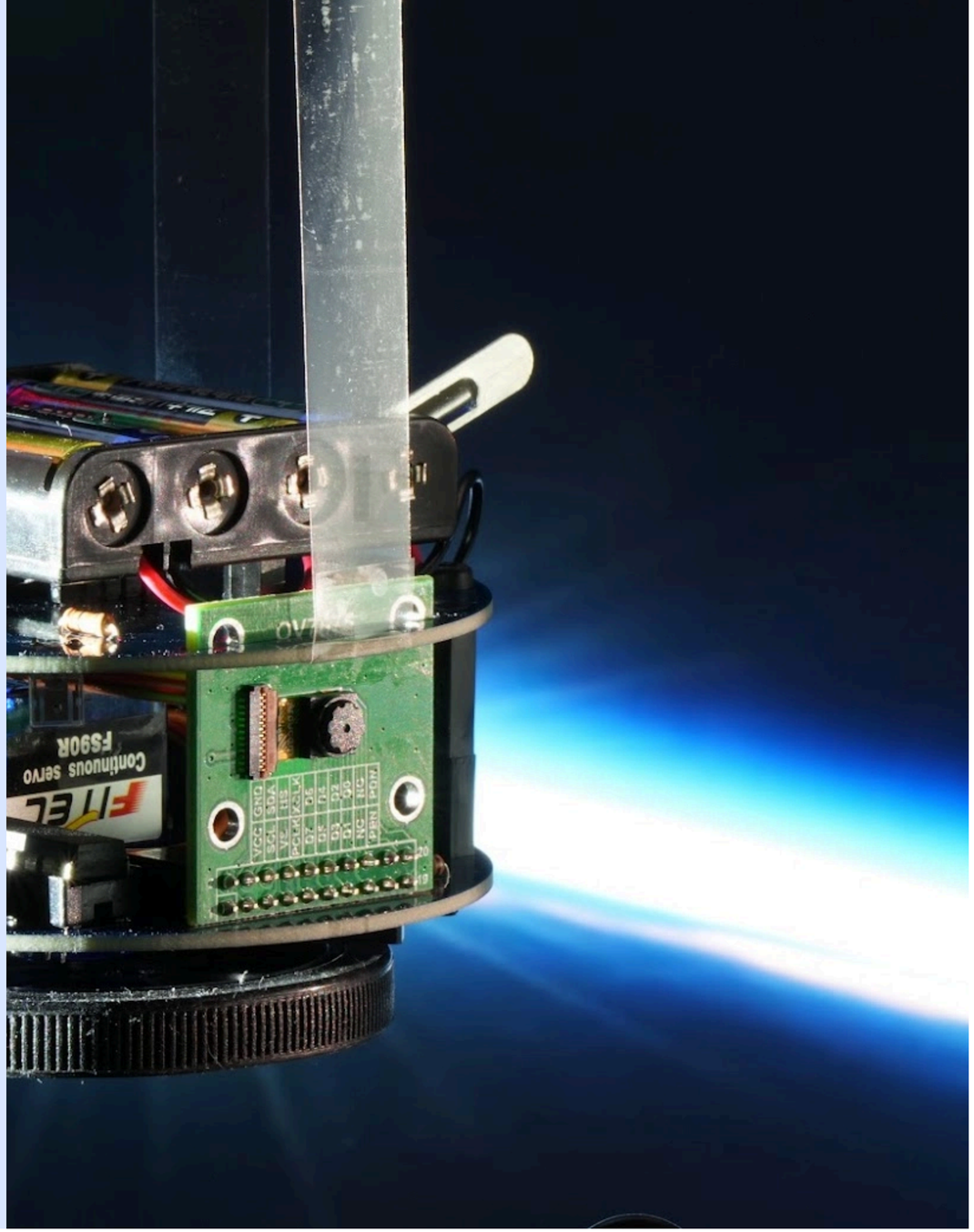
模擬衛星にも、宇宙機と同じ制約をかける

実衛星：物理が課す

手が届かないから、打ち上げたら触れず、電波でしか介入できない

模擬衛星：ルールとして課す

地上にあるが「運用中は覗かない、介入は電波を通してのみ」をルールにする



6.1 運用ルール

打ち上げは繰り返せる、でも上げたら覗けない

- **打ち上げ＝宇宙空間へ吊るして運用。**
  黒紙で囲った「宇宙空間」へ、衛星を吊るして運用すること
- **何度でも打ち上げてよい。**
  時間の許す限り（小型衛星ベンチャーのような高速イテレーション）
- **制約①：中は覗けない。**
  撮影した動画はゼミ終了後に閲覧可能
- **制約②：届くのは電波だけ。**
  観測・介入は地上局からの無線を通してのみ

機体が回るのを確かめられるのは、打ち上げの中だけ

開発中

- **機体は固定する**（自由回転で吊るすのは禁止）
- 回してよいのは**リアクションホイールだけ**
- 無重力は地上で再現できない

打ち上げ

- **吊るして自由回転**させ、姿勢を制御する
- 運用は**テレメトリだけ**（覗かない・電波だけ）
- 機体が実際に回るのを確かめられる唯一の場面

「開発中は回さないなら制御はどう作るのか」の手段は Week 7 で扱う。

6.3 今週の宿題

模擬衛星の初回打ち上げに必要な機能を考える

今日のまとめ：宇宙機の要求と制約から、その開発プロセスを理解すると、衛星の生存と観測可能性が重要だと分かる。

宿題の意図：それを模擬衛星に当てはめて具体的に考えてみよう。

注意

- 開発はチームで分担だが、**全員がシステム・開発プロセス全体に対して責任を持つ**
- そのために今回の宿題は、まず**自分1人でじっくり考える**（AIは積極的に使用してよい）
- 事前学習 Week 8 後のチーム内ミーティングで共有するまでは、それぞれの頭・パソコンの中で

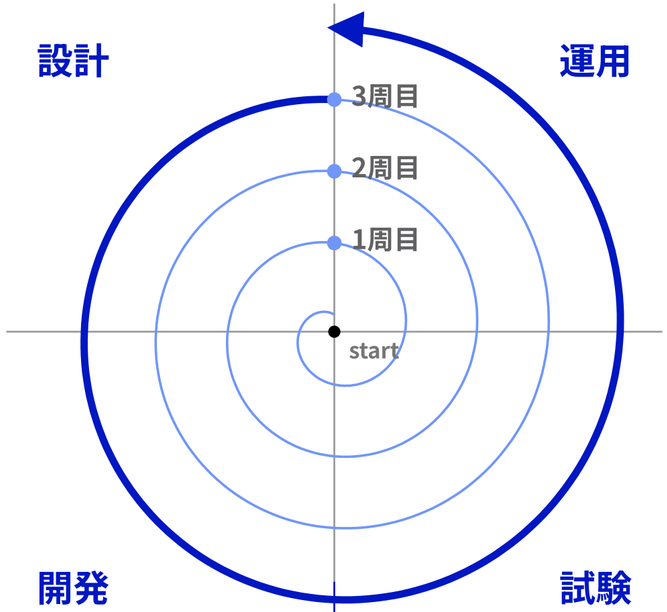
6.4 次回予告

Week 7　検証の仕組み化

テストとシミュレーションで、**誤りを人間より先に仕組みが見つける**状態を作る

同じサイクルを、地上で小さく何度も回す

- テストとシミュレーションで、**誤りを先に見つける**
- 内側の周は地上で完結する。**速く、何度でも回せる**
- 一番外側の周が打ち上げ。**手前で回した分だけ確からしさが上がる**



一番外側の1周＝打ち上げ

大きな「やってみる」

内側の周は地上でのテスト・シミュレーション。
同じサイクルの一番外側が打ち上げ。

付録・読み物 付録A この先は自分で調べる：備えの型の入口

講義が渡すのは名前と向きまで。中身と適用は自分で調べる

講義で扱う範囲

名前と向き

何のための型か。どちら向きの型か。ここまでは講義で渡した。

自分で調べる範囲

中身と適用

様式と手順、そして自分の衛星にどう当てはめるか。ここから先は各自で調べる。

注意

- この型が**自分の衛星に要るかの判断は自分でする**
- **宇宙の型をそのまま持ち込まない**。模擬衛星は運用が数時間で、放射線も真空も打上げ振動もない。同じ洗い出しから残るものが違う

壊れ方から影響へ辿り、起こりやすさと影響で対応を決める

FMEA

部品や機能の**壊れ方から影響へ辿る型**。FTA とは逆向き。

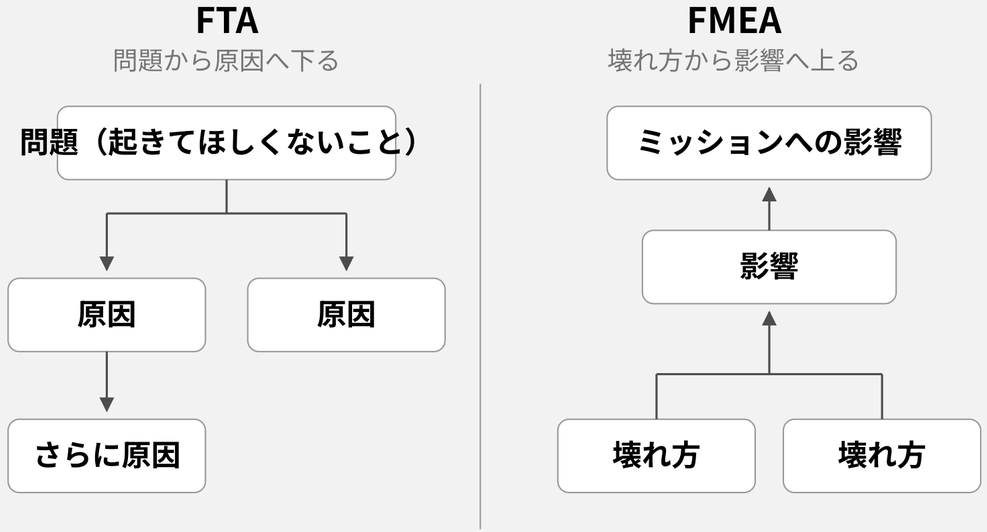
入口は「FMEA ワークシート」「RPN」。まずは3列（壊れ方／起きること／対策または確認方法）で足りる。

リスク管理

洗い出したものを**起こりやすさと影響で評価して対応を決める型**。

評価の根拠は感覚ではなく環境から取る。

備えないという判断も、根拠を言えるなら判断である。



FTA と FMEA は二方向の対。問題から下るか、壊れ方から上るか。

付録・読み物 付録C 異常を捕まえる型（HK テレメトリ・リミット）

起きうる壊れ方から、記録する項目と正常範囲を逆算する

HK テレメトリ

項目は**起きうる壊れ方から逆算して決める**。

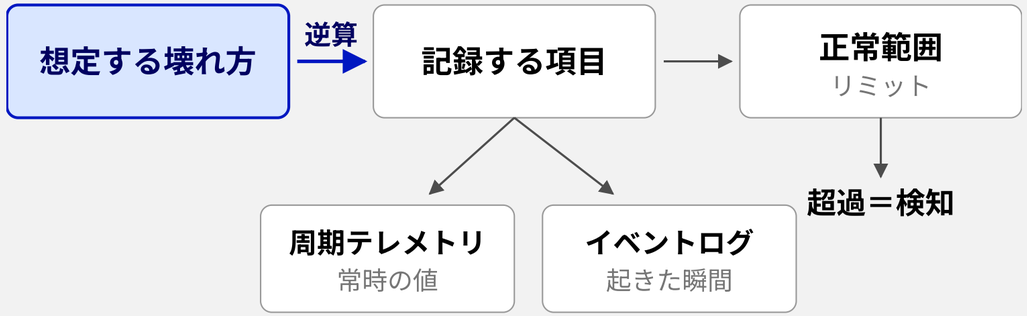
記録頻度は値の変化の速さに合わせる。常時の値は周期テレメトリ、起きた瞬間はイベントログ。

帯域と切り分け可能性のトレードオフで絞る。

リミット

各項目に**正常範囲を定めて超過を見張る**こと。これが検知になる。

閾値は勘で置かず、要求と制約から導く。



1つの壊れ方から、それを捕まえる項目と正常範囲が逆算で決まる。

付録・読み物 付録D 安全側へ逃がす型（セーフモード・ログ・介入用コマンド・FDIR）

検知・切り分け・復旧を、あらかじめ設計しておく

セーフモード

安全側に倒れた**既知の状態への遷移**（Week 4 の状態遷移の適用）。

何を止め、何を維持するかが設計。

ログ・介入用コマンド

ログは再起動やハングを**後から説明できる形**で残すこと。

介入用コマンドは地上から届く唯一の手段。

FDIR

検知・切り分け・復旧の枠組み。

環境が厳しくなれば、同じ考え方が冗長系・ウォッチドッグ・自律復旧の階層という厚い体系を要求する。

各段に対応する、模擬衛星での最小の型。

付録・読み物 付録E 覗けない相手へ手を届かせる型（OTA・コマンド体系）

届く経路は電波だけ。そこを通せるものを先に決めておく

OTA

無線経由のソフトウェア更新。

書き換えに**失敗したときに戻せるか**までが設計。

コマンド体系

地上から送れる操作の定義。

定義していない操作は**運用中に実行できない**。

何を可変にしておくか（閾値・周期・モード）が、そのまま運用中に打てる手の広さになる。