# ソフトウェア工学
# 第12回

土田　隼之

| 週 | 授業内容・方法 | 週ごとの到達目標 |
|---|---|---|
| 1週 | ソフトウェアの性質と開発 | ソフトウェア開発の特徴および課題について少なくとも一つ上げられ、その理由を言える。 |
| 2週 | ソフトウェア開発プロセス | 複数の開発プロセスモデルを挙げ、それぞれの特徴を言える。 |
| 3週 | 要求分析 | 要求分析とプロトタイピングの関係性や有用性について言える。 |
| 4週 | ソフトウェア設計 | モジュールの結合度の低い場合と高い場合のモジュール間の依存性について述べ、モジュール結合度の低い具体例を言える。 |
| 5週 | プログラミングとテスト | 誤り混入をさせないためのプログラミング手法およびテスト効率を向上させる技法について言える。 |
| 6週 | テストと保守 | 保守容易性を確保するための方策について、考察し、述べることができる。 |
| 7週 | グループワーク | 前半6週に関する課題を、グループワークで取り組む。 |
| 8週 | 中間試験 | 前半に習得した項目について確認する。 |
| 9週 | オブジェクト指向１ | 身の回りのモノに関して、クラスとインスタンスという言葉 |
| 10週 | オブジェクト指向２ | オブジェクト指向プログラミングの特徴について言える。 |
| 11週 | ソフトウェア再利用 | ソフトウェア再利用の重要性とその困難さについて言える。 |
| 12週 | プロジェクト管理 | プロジェクト管理の重要性を述べることができる。 |
| 13週 | 品質管理 | 品質管理手法について言える。 |
| 14週 | ソフトウェア開発規模と見積もり | ソフトウェア開発規模の見積もり手法について言える。 |
| 15週 | グループワーク | 後半6週に関する課題を、グループワークで取り組む。 |
| 16週 | 期末試験 | 後半に習得した項目について確認する。 |

ソフトウェア再利用も少し

模擬試験を予定してます

# 今日の内容

教科書の想定は「プログラムコード」など含めた「ソフトウェア部品」の再利用
→別プロジェクトのコード再利用は無くはないが、あまり無いかも

1)ソフトウェア再利用

・「ソフトウェア部品」の再利用について
・パッケージソフトの次バージョン開発は、良くある再利用
・linuxを使うことは、広い意味での再利用
→10年前、20年前よりも幅広く再利用されている

パッケージソフト、OSS戦略は企業方針の根幹
→就職を希望している企業は勝てる戦略がある？？
他社で出来ないことがあるから他社より待遇が良い
#例:再利用で効率的にソフト開発する
#例:利益率の良い顧客のシステムを作る

2)プロジェクト管理

SEにとっての評価規定みたいな位置づけでもある
#誰が、どんな内容を、いつまで行う
→どれくらい働けば相手(上司や取引先)は満足するのか
業務が問題なく進めばよいが、うまく進まなかった場合の調整材料

# 1)ソフトウェアの再利用

**ソフトウェアの再利用**:以前に開発した成果を利用することにより、ソフトウェアを開発する手法のこと。

具体的には、ソフトウェア開発に必要とされる知識、プログラムやデータをなんらかの形でパターン化、標準化、部品化することにより、繰返し利用を可能にする。既存ソフトへの新規機能追加や、既存機能の変更も含む。

→開発期間の短縮などの生産性向上や、すでに品質保証済みの部品を使うことによる信頼性・保守性などの品質向上

**例:DBMSのSQL解析部分を、自社の他DBMSのものを再利用**

**品質保証済であるのは大きい**

→品質保証にかかるコストは思ったより大きい

ファーストユーザー(例:新規開発されたDBMSパッケージソフトの初回利用)は嫌がられることも

実績重視(客側は、不必要なリスクは負いたくないので。。)

# ソフトウェアの再利用の考慮事項

ソフトウェア再利用においては、以下の項目を十分考慮して進めていくことが重要である。

## a)再利用効果の現れ方

ソフトウェア再利用の方策は、すぐに効果のでるものではない。再利用しようとする対象ソフトや関連ツールなどの理解のために、開発効率が低下することも。**→何でもかんでも再利用するわけではない。**

**例:OSSを組み込んだ製品で、OSS不具合出ると対応してくれるかはコミュニティ次第。。**

## b)技術的アプローチ以外の事項の重要性

ソフトウェア再利用は、技術的なアプローチだけではうまくいかない。再利用のための支援組織の確保や教育訓練の実施など、組織全体としての推進体制、再利用ソフトウェアの契約上の問題、その分野の安定性など、非技術的なアプローチを合わせた方策が重要である。

**→Redhatだとかにエンジニアを送り込んでいるケースも**

HITACHI
Inspire the Next

検索 | Japan | 日立グループの製品・サービス | 日立グループの企業情報

ソフトウェア | プラットフォームソフトウェア

Global Site | サイトマップ | お問い合わせ

製品 | サポート | 事例紹介 | セミナー＆イベント | 新着情報 | かんたん検索

サイトトップ > 製品 > Linux Information > トピックス&ニュース

# Linux Information：トピックス&ニュース

2021年 2020年 2019年 2018年 2017年 2016年 2015年 2014年
2013年

## 2021年

**2021年9月30日**
- Red Hat Enterprise Linux Server 6、Red Hat Enterprise Linux Server 7、Red Hat Enterprise Linux Server 8の動作確認状況を更新しました。

**2021年8月31日**
- Red Hat Enterprise Linux Server 7、Red Hat Enterprise Linux Server 8の動作確認状況を更新しました。
- Red Hat Enterprise Linux Server 8の出荷品情報を更新しました。
- 次のカタログを更新しました。
  - Red Hat Enterprise Linux バージョン固定保守オプション

**2021年7月30日**
- Red Hat Enterprise Linux Server 7、Red Hat Enterprise Linux Server 8の動作確認状況を更新しました。
- Red Hat Enterprise Linux Server 8の出荷品情報を更新しました。

**2021年6月30日**
- Red Hat Enterprise Linux Server 6、Red Hat Enterprise Linux Server 7、Red Hat Enterprise Linux Server 8の動作確認状況を更新しました。
- Red Hat Enterprise Linux Server 7の出荷品情報を更新しました。

Linux Information
- 製品紹介
- サービス紹介
- 紹介記事
- カタログ
- 製品お問い合わせ

お客さまの360°（全方位）を日立の360°（総合力）で守ります


- お客さまシステムの多面的

**RedHatを用いたシステム開発するには、不具合出た場合に対応できる必要**

https://www.hitachi.co.jp/Prod/comp/soft1/linux/topics/index.html

Qiita Zine

ホーム

インタビュー、ダイアップ 日立のエンジニアライフの実態とは？超エンジニアドリブンな…

23
8月
2021

インタビュー　ダイアップ

# 日立のエンジニアライフの実態とは？超エンジニアドリブンな、OSSセンタのワークスタイルに迫る。

Tweet　Share　B! Hatena　Pocket　RSS


社会の共有財産として、コミュニティドリブンな貢献によって進化してきたオープンソースソフトウェア（以下、OSS）。デジタル分野での先進機能はOSSによって実現されるケースが増えており、企業においても先進的な技術によるメリットを享受するためにOSSが広く活用されています。

**――素朴な疑問なのですが、扱うOSSって、どんな基準で決めているのですか？**

**茂木：**今、そこでちょうど悩んでいるところです。
基本的には、できるだけ広く使ってもらうものでなければいけないのですが、ビジネスにおいて実際に組む人が本当に必要なものか、ライセンスは問題ないかなどについても悩むわけです。

**――悩んだ結果、最終的にはどうされているのですか？**

**茂木：**結局は「面白そうだよね」が決め手かもしれません（笑）

**現在使われてないOSSも、10年後は使われているかも→会社でも調査**

**――いいですね（笑）最近「面白い」と感じたOSSはありますか？**

**茂木：**最近見ているものだと、ポリシー管理ツールである、「Open Policy Agent(OPA)」が面白いですね。Kubernetesの設定検証などによく使われるものですが、APIアクセスの認証認可の検証で使えばAPI Gatewayの処理から認可処理をOPAに分離できるため、異なる種類のAPI Gatewayで認可制御のロジックを再利用できるなどのメリットがあります。

**――面白いですね！新しいOSSを取り入れる際に難しいなと思われるポイントはありますか？**

**茂木：**新しいOSSを見つけたとして、それの継続性ですね。ライセンスなどの観点含め、果たして会社の中で使っていって良いものなのか、コミュニティの開発が止まってしまわないか。技術のトレンドは移り変わりが早いし、人の育成も必要だし、お金もかかる話なので、ちゃんと継続して使えるものになるのか、価値を生み出せるものなのか判断は、本当に難しいところです。

https://zine.qiita.com/interview/202108-hitachi-2/?fbclid=IwAR1mWcs31FWOg4wJu8tQtrrlcclu_pG_R_NphcWTXsJ2vU-FxbNAnsivFxQ

## コミュニティ貢献・標準化活動

日立ではメインフレームからはじまる長年のプラットフォーム開発と顧客サポートの経験を生かし、1990年代後半からLinuxカーネルの高信頼化に貢献してきました。特にThe Linux Foundation(LF)*では、創立当初からスポンサーとしてコミュニティのサポートや開発貢献、イベントの運営等に取り組み、OSSの普及促進に尽力してきました。 現在はThe Linux FoundationのPlatinum Memberとして、積極的な貢献をしています。近年は、The Linux FoundationのHyperledgerに加盟し(2016年2月)、共同開発に参画しています。 その他、クラウドプラットフォームとしての期待が大きなOpenStackをはじめ、さまざまなコミュニティ・標準化団体に参画し、パッチ投稿やセミナーでの発表等を行っています。

* The Linux Foundation：Linuxを中心としたOSSの開発コミュニティや開発者のサポートを行い、OSSの発展を支えている世界随一の団体(本部はUS)。 "Linux"商標の保護を柱としたOSSの法的支援や、世界のLinux開発者が集うOpen Source Summit(旧LinuxCon)、Embedded LinuxCon等のカンファレンスも運営している。

参画している主な団体 貢献者のご紹介 セミナーでの発表 ウェブサイトへの掲載 その他

### 参画している主な団体

| 分類 | 団体名称 | 日立における最近の主な活動 |
|---|---|---|
| 全般 | The Linux Foundation | • Platinum Sponsorの理事として、運営方針の策定等を実施。<br>• OSS共同開発Projectへの参加：<br>• OpenChain Project<br>• プラチナメンバーとして活動中。<br>• OpenChain v2.1 (ISO/IEC 5230)仕様適合の認証を取得。<br>・OpenChain プロジェクトからのアナウンス<br>• OpenChain仕様適合の第三者認証を取得。<br>・テュフズードジャパンのプレスリリース<br>・OpenChain プロジェクトからのアナウンス<br>• Hyperledger Project<br>• ボードメンバー(Premier)として参加。各種機能の開発に貢献中。<br>• Hyperledgerが認定するベンダー資格 Hyperledger Certified Service Provider（HCSP） を取得。<br>• CIP(Civil Infrastructure Project)<br>• 設立メンバー(Silver)として参加。<br>• セキュリティWGに参画。<br>• 産業界でLinuxの長期サポートを提供すべく活動中。<br>• AGL(Automotive Grade Linux)<br>• Bronzeメンバーとして参加。 |

**コミュニティで活動してないと、要望が反映されない**

https://www.hitachi.co.jp/products/it/oss/efforts/index.html#community

# ソフトウェアの再利用の考慮事項2

ソフトウェア再利用においては、以下の項目を十分考慮して進めていくことが重要である。

**c)再利用されるソフトウェア品質の判定**

ソフトウェアの品質は、再利用しようとするソフトウェアの品質に大きく依存する。そのため、再利用対象ソフトウェアについての信頼性、使用性、効率性などの品質が確保されているかで、使用可能なソフトウェアか判断する必要がある。

例:使用実績があり信頼性が確認されているか、メモリサイズや処理速度などの性能が要求と合っているか

**メジャーなOSSを使うのはOKとして、どこまではOK?**
**例:Linux、GCCは良いとして、、**

# ソフトウェアの再利用の課題

現実には、いくつかの課題により、再利用は簡単には実現できない。

**a)下位レベルの再利用は効果が小さい**

小さいプログラム部品などの下位レベルの再利用では、削減できる作業量より、再利用のための追加作業量が多くなる。

**b)上位レベルの再利用は標準化が難しい**

上位レベルのプログラム部品や仕様は多様性が大きく、共通化や標準化が難しい場合が多い。→銀行(地銀)がシステム共同利用

**c)応用分野ごとに標準化が異なる**

ある応用分野で標準化した部品ソフトが蓄積されても、分野が異なれば新たな標準化作業を行う必要がある。作業量がむしろ増加してしまうことも。

# 横浜銀行など地銀5行とNTTデータ、共同システムのオープン化で合意

山端 宏実　日経クロステック／日経コンピュータ

2021.04.01

PR
2人がかりだったサーバーの入れ替えが1人で簡単にできるサーバーとは？
過酷を極める作業現場にデジタルの恩恵、DXを一歩先へ進めるキーアイテム
DXは転職のチャンス！まずは自分の転職市場価値を確認/エグゼクティブ転職

横浜銀行や七十七銀行など地銀5行とNTTデータは2021年4月1日、勘定系の共同利用システムについて、オープン基盤の採用で合意し基本契約を締結したと発表した。現行の共同利用システムはメインフレームで動作しているが、2024年をめどにLinuxを採用したオープン基盤へ移行する。将来的にクラウド移行も視野に入れる。

NTTデータによると、勘定系の共同利用システムのオープン化は銀行業界で初めて。参画する銀行は2行のほか北陸銀行、北海道銀行、東日本銀行。5行は2010年から順次、NTTデータが開発・運用する「MEJAR（メジャー）」と呼ばれる共同利用システムを使用。勘定系のほかATMやインターネットバンキングなども5行で共同

**銀行業務システムを共同利用**
**Linuxを採用したオープン基盤**
**→メインフレームでない**

https://xtech.nikkei.com/atcl/nxt/news/18/10016/


**応用ソフト(ソリューション)を他分野にも横展開したい。**
**→ソフトは一回作れば、販売量が増えても原価はあまり変わらない**
**→MSやOracleは儲かってる。**

https://www.hitachi.co.jp/products/it/lumada/solution/index.html

# 再利用によりソフトウェアを開発する手法1

再利用してソフトウェアを開発する手法には、応用プログラムの再利用、あるいはフレームワーク、パッケージソフトウェアなどを用いるものがある。

## a)応用プログラムの再利用

開発しようとするソフトと類似のプログラムがあるか探す。該当するプログラムが見つかれば、仕様書、ソースコードを調査・分析して、再利用可能な部分を切り出す。

**データベースはパッケージソフトの一種**

## b)パッケージソフトウェア

パッケージソフトは、外部から購入・導入する完成度の高いプログラム部品である。あらかじめ仕様の変更範囲が設計されており、その範囲内で対応するようになっている。導入の留意点として、例えば下記がある。

①機能や品質の事前確認、②開発ベンダや契約内容の確認、③利用技術の習得、④保守と保証の確認

# 再利用によりソフトウェアを開発する手法2

**c)フレームワーク**

フレームワークとは、カスタマイズ可能なようにソフトウェア部品を前もって組み立てた半完成ソフトウェアである。利用にあたっては、アプリケーションをフレームワークに適合させる必要があり、以下の課題がある。

・設計の自由度が小さくなる

・フレームワークの選択と設計が重要になる

**ソフトを作成する手助けをする手助けをするパッケージソフトの場合も**

# フレームワークとは

フレームワーク:あるアプリケーション分野の基本構造と、基本機能を持つプログラムライブラリのこと

**→再利用可能なコードをフレームワークにまとめることにより、新たなアプリケーションのために同じようなコードを改めて書かなくて済むようになる。**

**例:並列実行部分はフレームワーク
実行関数はユーザが新規作成**

**・ライブラリとの違い**

ライブラリを用いる再利用は、アプリケーションがインタフェース層を介してライブラリを呼び出す形態となる。アプリケーションが制御の主体であり、必要なときに必要なライブラリを呼び出す。

**例:圧縮ライブラリ(gzip)をDBMS内で使ってデータ圧縮**

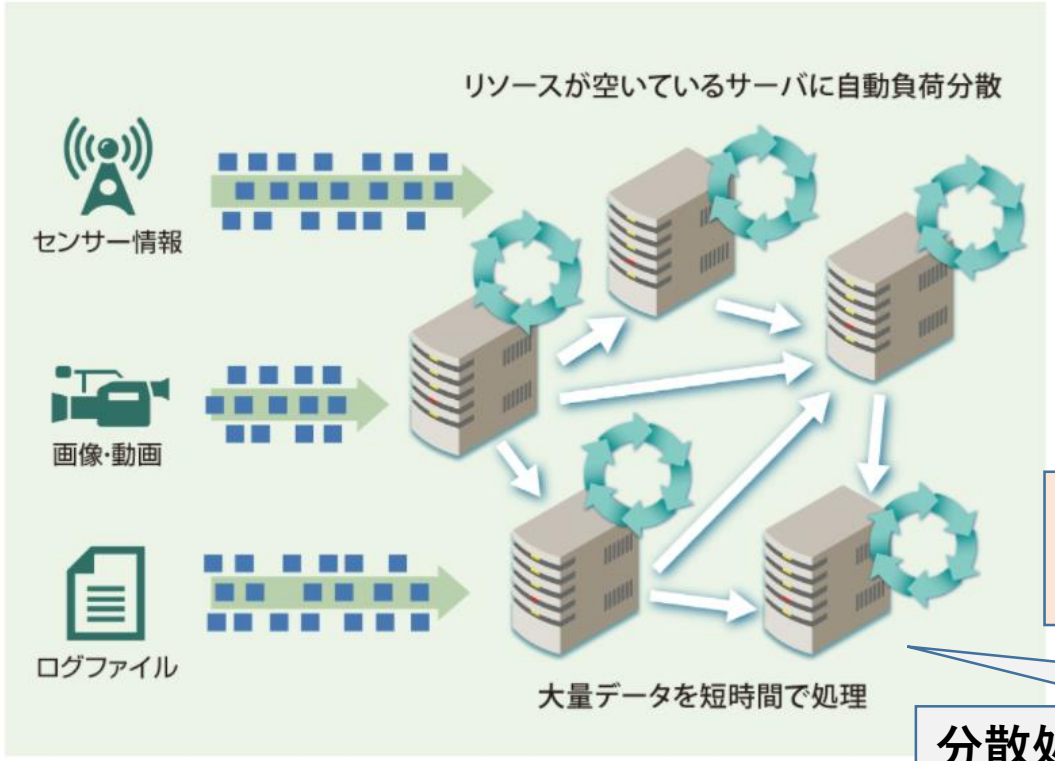
ライブラリは機能的モジュールの部品化であり、制御構造自体を部品化するものではないこと、部品の入出力データ構造も限定されるなどから、**文字列処理や数学的関数群などのような汎用部品しか有効な再利用例はなかった。**

# 制御型フレームワーク

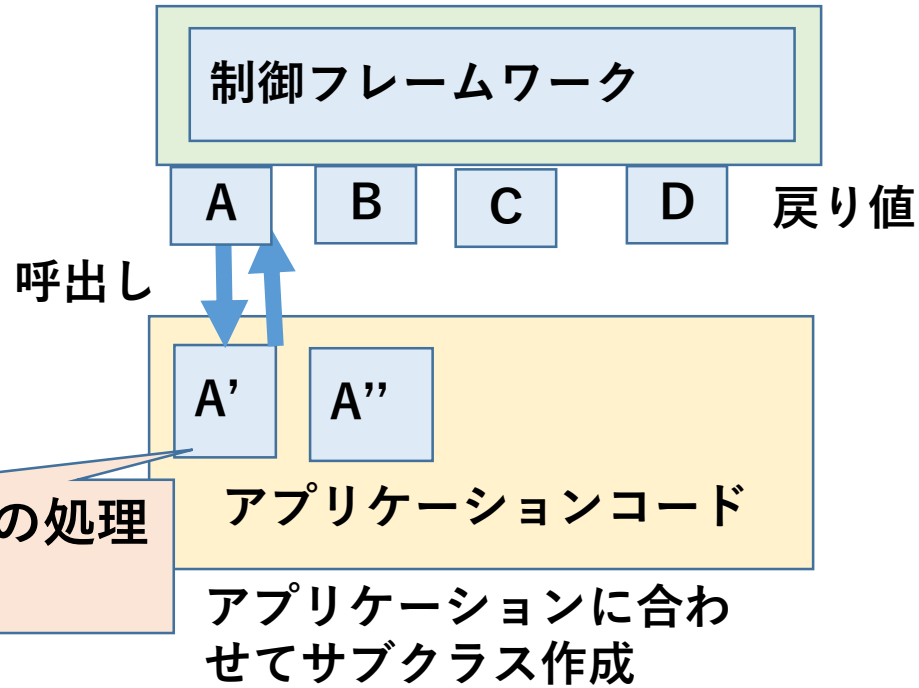
制御型フレームワークは、フレームワークが制御の主体となり、アプリケーションコードを呼び出すタイプの再利用形態である。アプリケーションコードが制御を得るのは、フレームワークがアプリケーションコードを呼び出した時だけである。

**【 素早く"分かる" 】大量データを高速に分散処理でき、分析・検証結果がすぐに分かります。**

大量データやAIを用いた高負荷な分析を、複数のサーバで効率的に分散処理します。受け取ったデータを、リソースが空いているサーバに自動負荷分散することで、処理時間を短縮可能です。 分析・検証結果がすぐに分かるため、データの価値化、新サービスの創出を加速します。




**制御の主体**
**【例】A→B→C→Dを繰り返す**



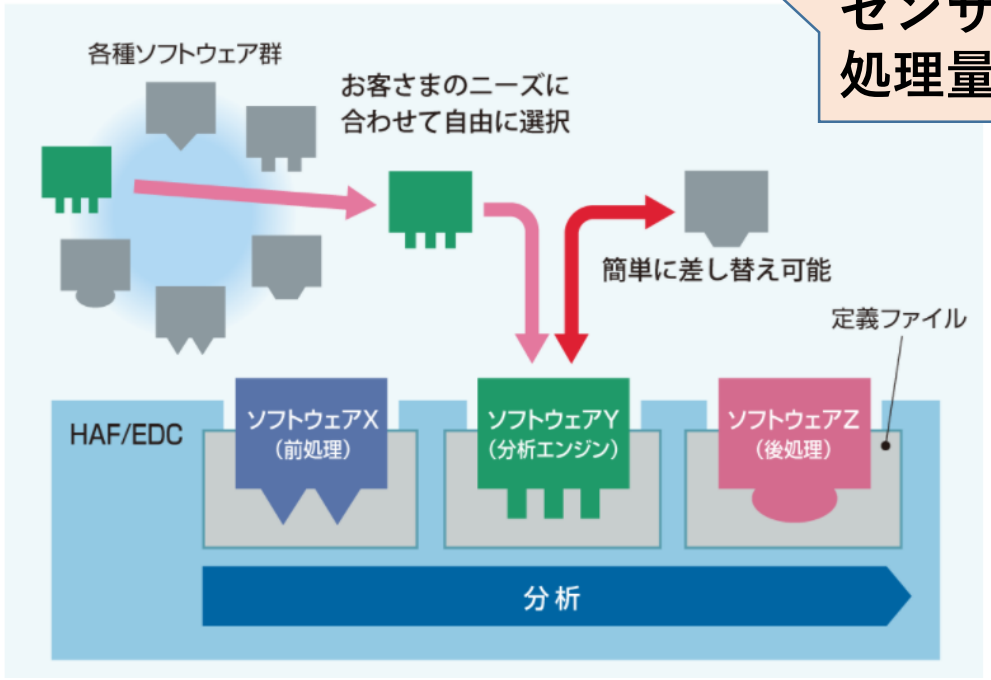

**届いたデータ毎の処理**
**例:異常検知**

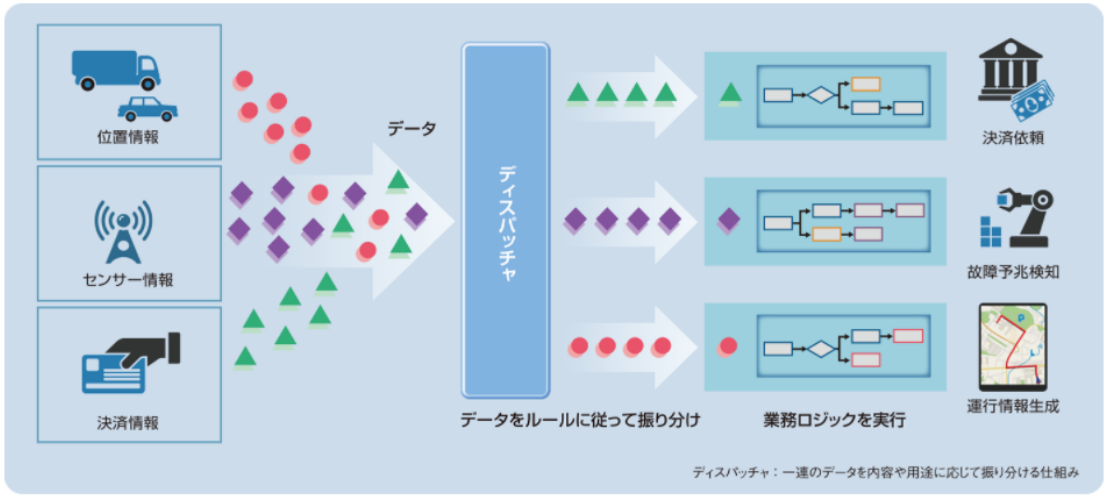
**アプリケーションに合わせてサブクラス作成**

**分散処理する内容だけを実装して、フレームワークに組み込む**

【 手軽に"試せる" 】IoTに適したフレームワークを活用することで、仮説検証を容易に行えます。

IoTに必要な「格納/分析/参照」のプロセスをフレームワーク化し、データの格納や分析に必要な各種ソフトウェア群を連携させる実行基盤を提供します。 極力"作らない"コンセプトにより、ビジネス要件の変更を柔軟に取り込みつつ、短期間でのシステム構築を実現します。

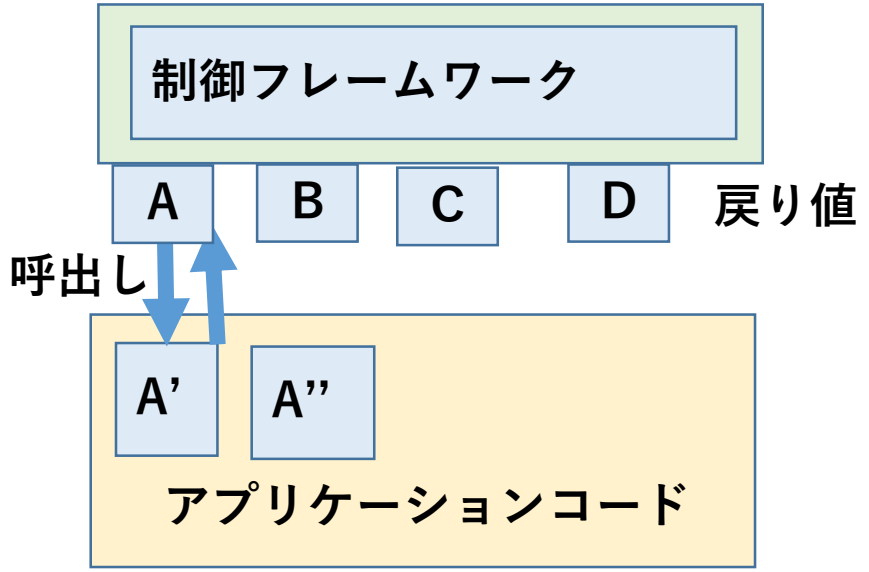



**センサデータ(IoTデータ)は、処理量の増減が一般に大きい**

**制御の主体**

**【例】A→B→C→Dを繰り返す**

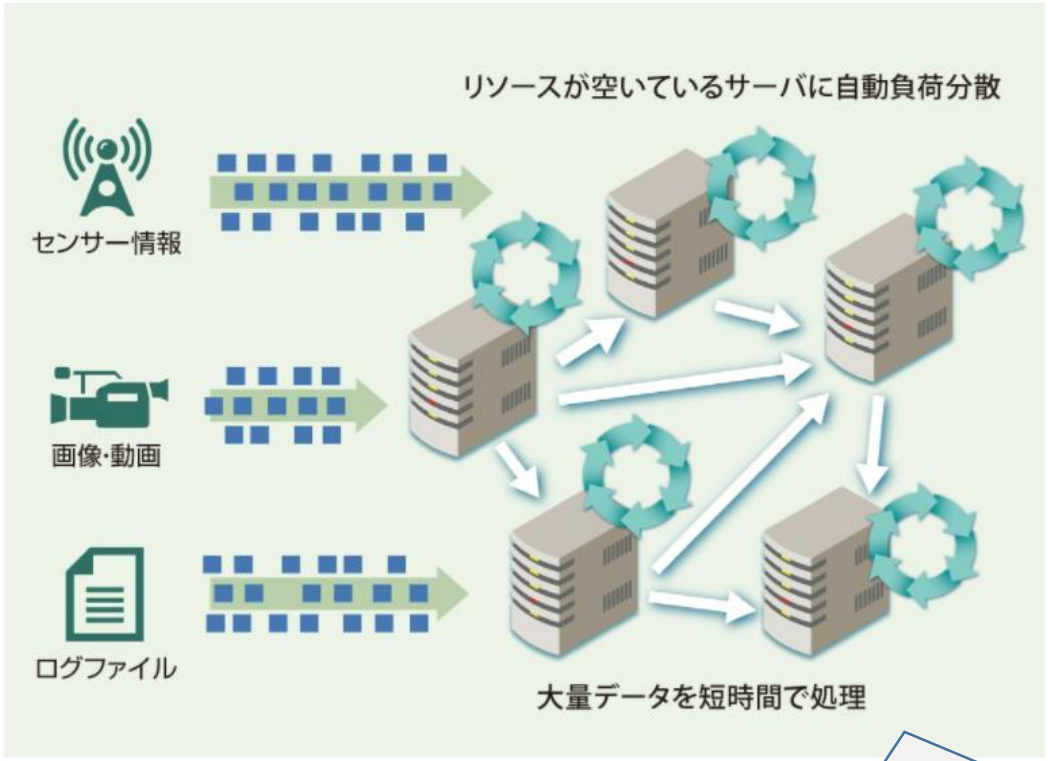



**アプリケーションに合わせてサブクラス作成**

https://www.hitachi.co.jp/products/it/society/product_solution/telecom/infrastructure/haf.html

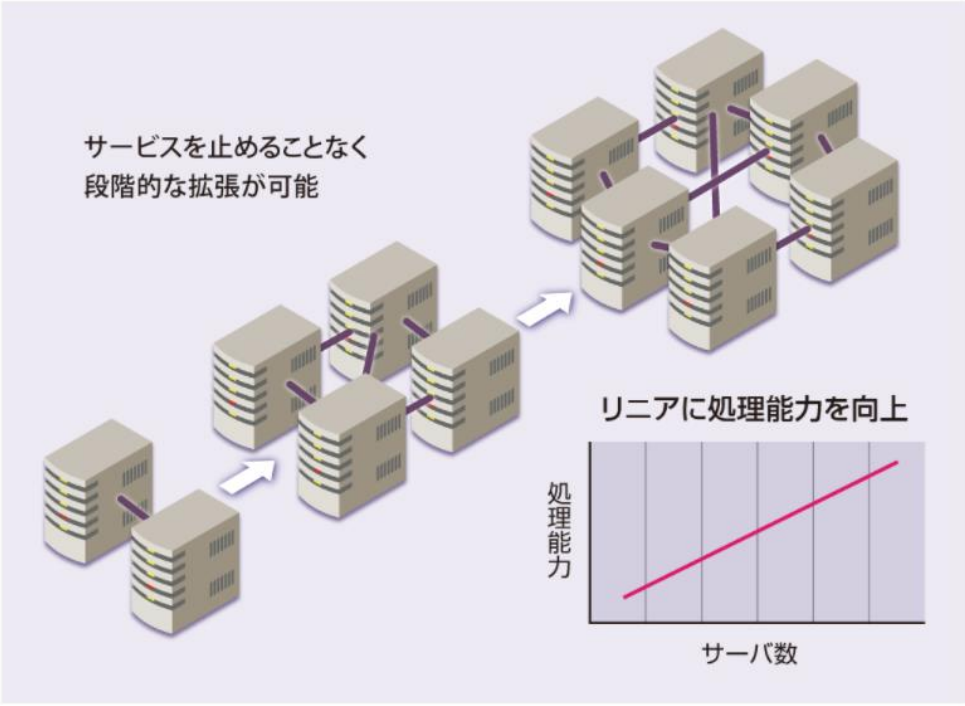
**【 素早く"分かる" 】大量データを高速に分散処理でき、分析・検証結果がすぐに分かります。**

大量データやAIを用いた高負荷な分析を、複数のサーバで効率的に分散処理します。受け取ったデータを、リソースが空いているサーバに自動負荷分散することで、処理時間を短縮可能です。 分析・検証結果がすぐに分かるため、データの価値化、新サービスの創出を加速します。




**【 手間なく"増やせる" 】処理するデータ量に合わせて、シームレスにシステムを拡張できます。**

データの規模が拡大する際にも、サーバを増やすことで、サービスを止めることなく簡単にシステムの処理能力を向上できます。 自動で構成変更するアーキテクチャにより、スモールスタートで仮説検証に取り組み、そのままスムーズに商用システムへ移行できるため、初期投資と運用コストを抑えることが可能です。




**分散処理機能を作らずに済む**

# 第12-1回課題(締切7/21)

課題1)就職志望を考えている企業(or業界)が利用しているOSSにどのようなものがあるかを調べて、100～200文字で概要をまとめてください。

課題2)隣の人とまとめた内容を見せ合ってお互いに感想を共有してください。

提出:課題1)でまとめた文章、課題2)相手からの感想(50文字程度)

# 今日の内容

1)ソフトウェア再利用

・「ソフトウェア部品」の再利用について

・パッケージソフトの次バージョン開発は、良くある再利用

・linuxを使うことは、広い意味での再利用

→10年前、20年前よりも幅広く再利用されている

「普通の(当然に想定される)成果」は人によって違う
→業務の方向性なども調整

2)プロジェクト管理

SEにとっての評価規定みたいな位置づけでもある

#誰が、どんな内容を、いつまで行う

→どれくらい働けば相手(上司や取引先)は満足するのか

業務が問題なく進めばよいが、うまく進まなかった場合の調整材料

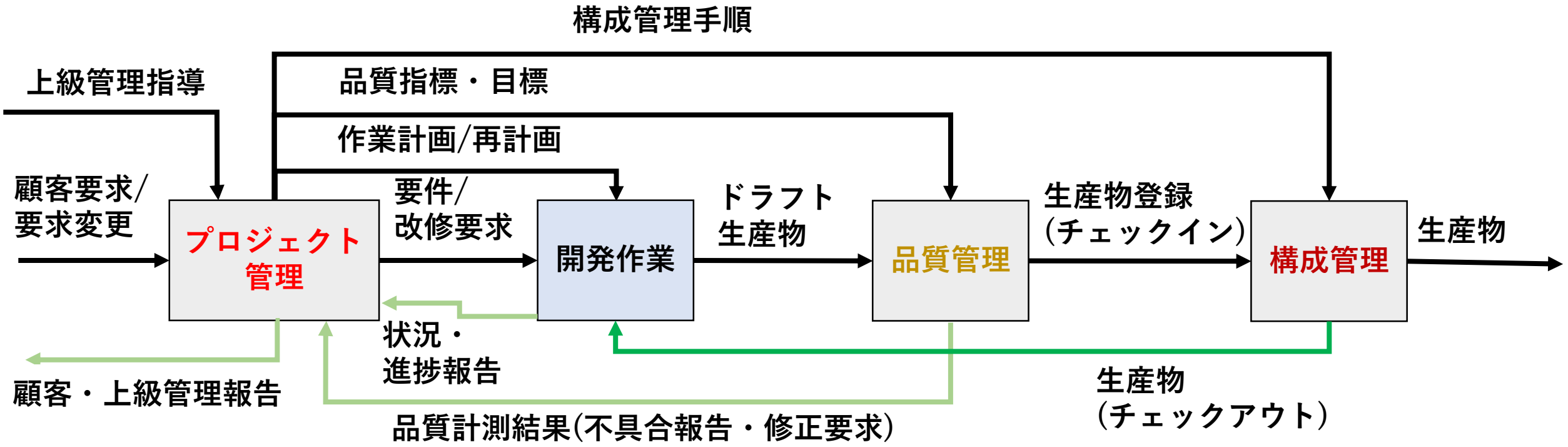
# プロジェクト管理と品質管理

大規模なソフトウェア開発を効率的に実施し、かつ一定の品質を達成するためにはソフト開発管理が重要である。

**プロジェクト管理**:ソフトウェア開発自体を計画・計測・制御する組織的活動

**品質管理**:ソフトウェア製品と開発プロセスの品質を計画・計測・制御する組織的活動

**構成管理**:ソフトウェア製品の一貫性を保持する活動



# 開発管理の枠組み①

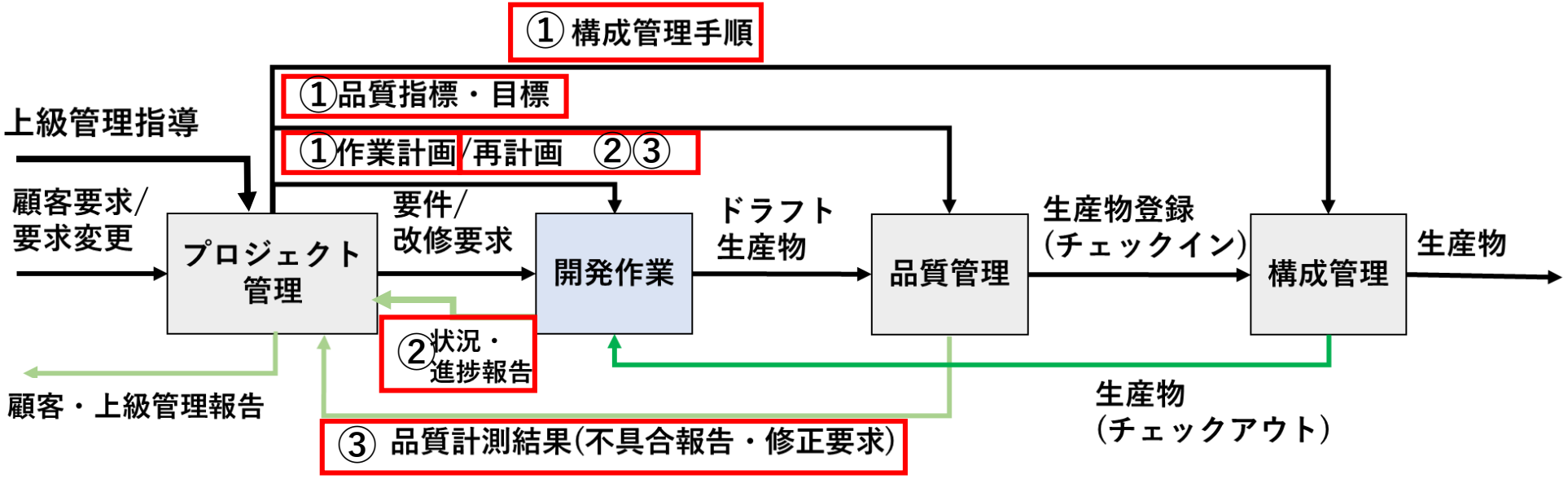
大規模なソフト開発は、複数の人間が組織的に活動してはじめて可能となる。これを効率良く行うためのマクロな仕組みが開発管理である。

**開発管理の要素:**プロジェクト管理、品質管理、構成管理

**プロジェクト管理:**①顧客要求を入力として作業計画、品質指標・目標、構成管理手順を策定する。

②開発開始後は、開発作業の状況・進捗報告をモニタし、問題があれば開発作業に対して再計画を行う。

③品質測定結果がNGの場合にも開発作業に対して再作業を指示する。



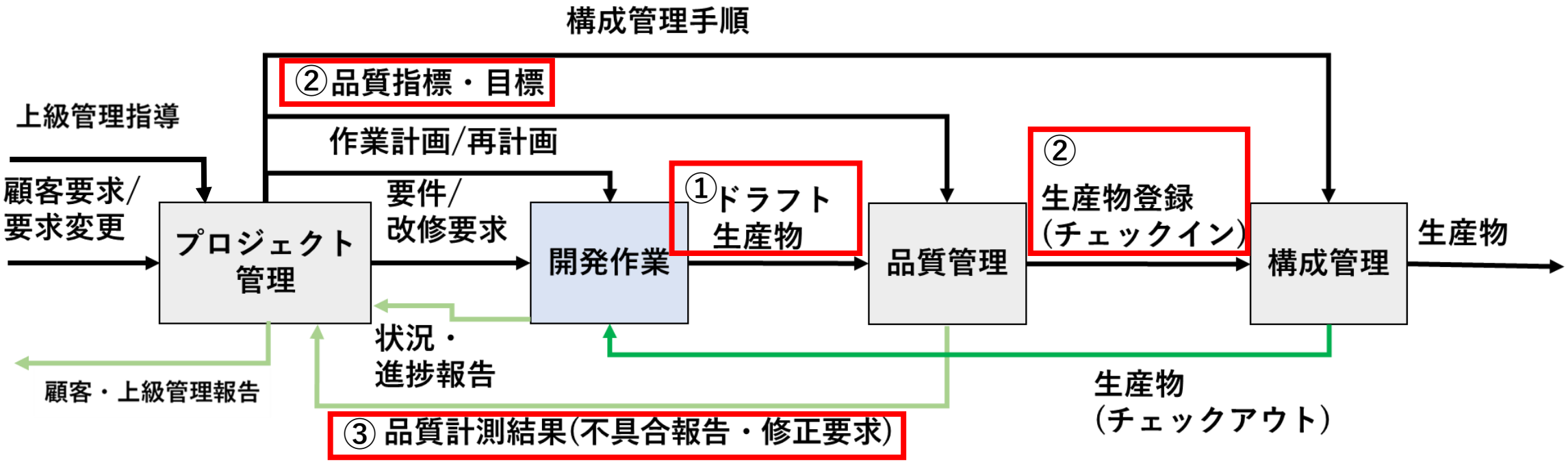
# 開発管理の枠組み②

**開発管理の要素:**プロジェクト管理、品質管理、構成管理

**品質管理:**①開発作業の結果としてのドラフト生産物を受け、

②プロジェクト管理であらかじめ計画された品質指標と品質目標を入力として、その品質を検査する。ドラフト生産物が品質目標を満たせば生産物として登録(チェックイン)し、そうでない場合には登録を拒否する。

③品質計測の結果は、プロジェクト管理に報告される。



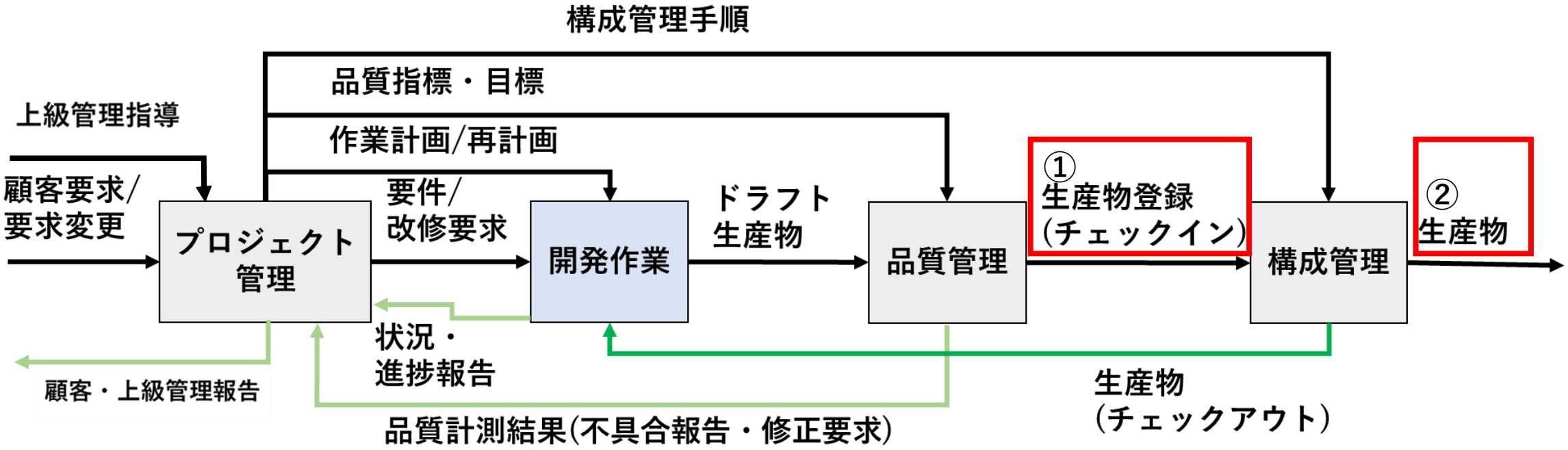
# 開発管理の枠組み③

**開発管理の要素:**プロジェクト管理、品質管理、構成管理

**構成管理:**

①所定の品質を満たした生産物を登録(チェックイン)し、

②要求変更や不具合修正に伴う修正変更を行えるように生産物を開発作業に引き渡す(チェックアウト)役割を持つ。

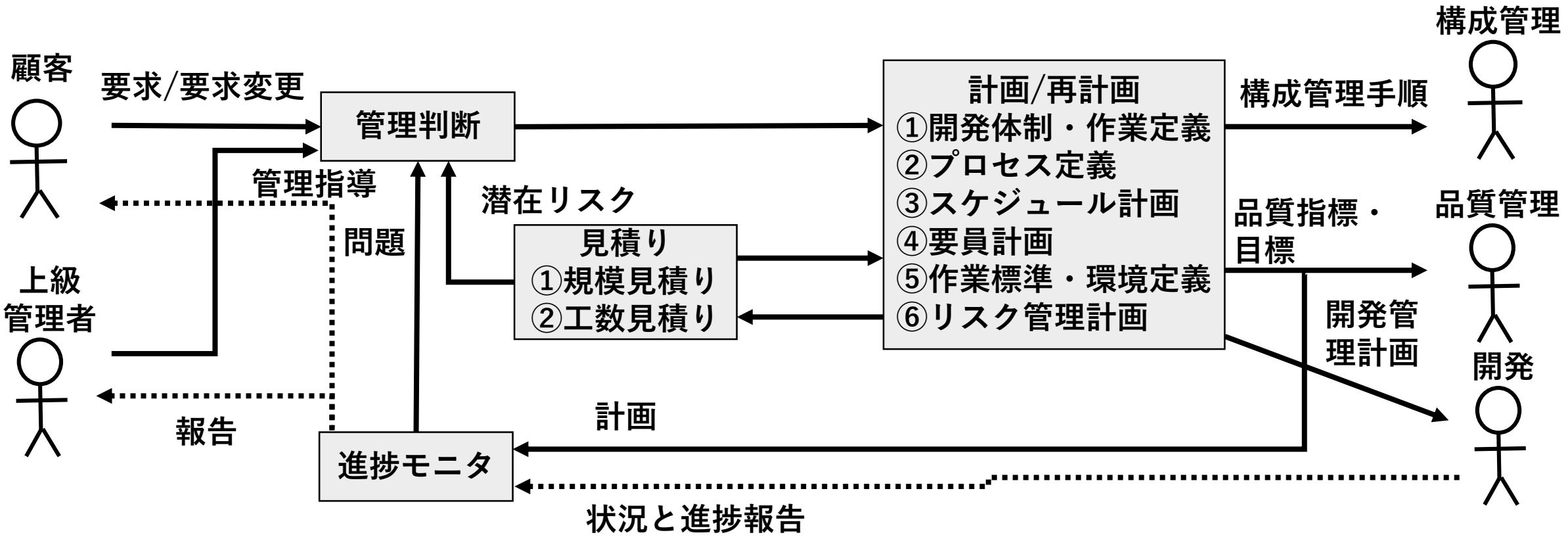


# プロジェクト管理

ソフトウェア開発組織は、ソフトウェア開発作業を、要求されたコストと期間および品質を満たすように制御していく必要がある。

**プロジェクト管理:**ソフト開発に必要なあらゆる面に関して計画を立て、開発の実態がその計画内に収まるように行う制御活動

**プロジェクト管理の構成要素:**見積り、計画/再計画、進捗モニタ、問題識別、管理判断



# 見積り

ソフトウェア規模の見積り、開発工数の見積りがある。

**ソフトウェア規模見積り:**開発するソフトウェアに対する要件から、ソフトウェアそのものの規模を推定すること

#ファンクションポイント数(次週以降扱う)、ソースコードライン数で表す

**開発工数の見積り:**ソフトウェア規模見積りをベースとして、使用する開発プロセス・言語・環境、組織自体の開発能力、ソフトウェアの再利用率などから、開発にかかる工数を推定すること

#マンパワー(開発人員×開発時間)で表す

プロジェクト管理においては、開発工数の見積りが非常に重要

→見積り工数がプロジェクト計画のための根拠となる

　開発工数が過度に低く見積もられれば、本来必要な開発リソースが投入されない。最良の管理行動をとったとしても、プロジェクトが破綻してしまうことも。

# 計画/再計画

**完全な計画を作成するのは不可能であるが、何を、誰が、いつまでに、どのように行うのかを決める必要がある。**

#プロジェクトは、程度の差こそあれ新規開発部分が必ず存在するので、行うべき作業とその見積り工数がすべてわかっていることはない。そうであっても、実績は計画の差異で把握され、計画作成は必須となる。

基本的な留意点:

①不明なことは残しつつ、作業は人が行える程度まで詳細化する

②組織標準、類似プロジェクト情報など過去の知識を基準に、スケジュール、リソースなどの制約を満たすようにカスタマイズ

③チームと個人の役割責務とコミュニケーション手段を明確に

④あらゆる種類のリスクを考慮しその対処手段を計画する

⑤要求変更、スケジュール遅れなど種々の事態に対処できる仕組みを作る

# 開発体制・作業定義

**作業定義:**成果物を生産するうえで必要な全作業要素を定義し、その各作業要素の工数見積りを行うことで、WBS(Work Breakdown Structure)が使われる。

**WBS:**各階層レベルで管理対象が明確かつ要求される精度の工数見積りを行えるように構成する必要がある。

各階層の見積り精度は、開発予算獲得のレベル、外部組織へのタスクの委託をできるレベル、人員が担当チームに割り当てられ人/日が予測できるレベルがある。

**開発体制定義:**プロジェクト組織内におけるグループやチームに対して、役割、責任および報告関係を文章によって定義すること。

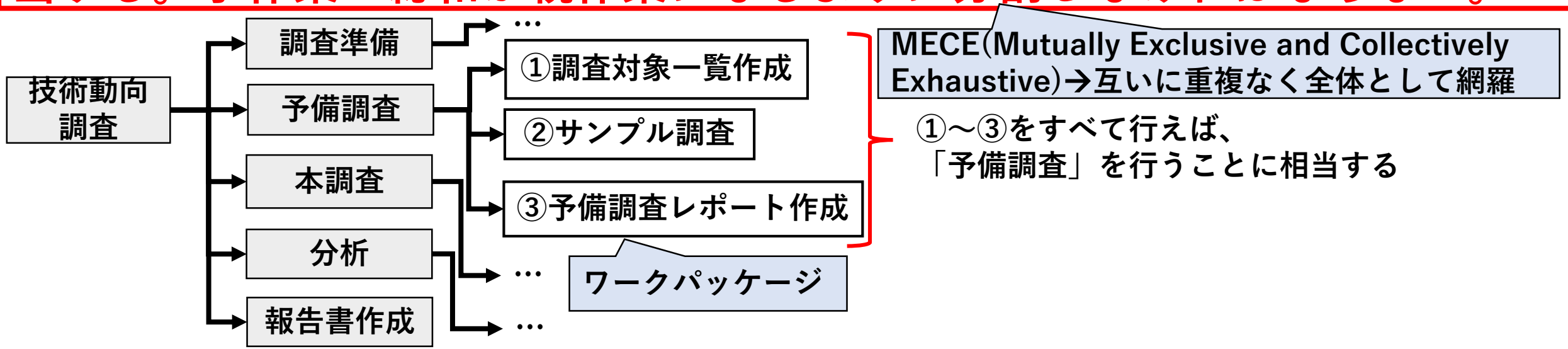
# WBS

プロジェクトでは、最初は大まかに物事が決まり、それが徐々に明確化、詳細化されていくという特性がある。WBSでは、必要な成果物を作るために実行する作業を階層的に要素分解する。

A)プロジェクト全体の作業とそれを達成するために必要となるより小さい作業とが体系づけられる。

B)末端でこれ以上分割されない作業をワーク・パッケージと呼び、時間やコストを管理する際の基本単位となる。

**C)分割時には、下位の作業をすべて行うことが上位の作業を行うことに相当する。子作業の総和が親作業になるように分割しなければならない。**



# WBSの作成

WBSでは、必要な成果物を作るために実行する作業を階層的に要素分解する。

①トップダウンで作業を分解ください。

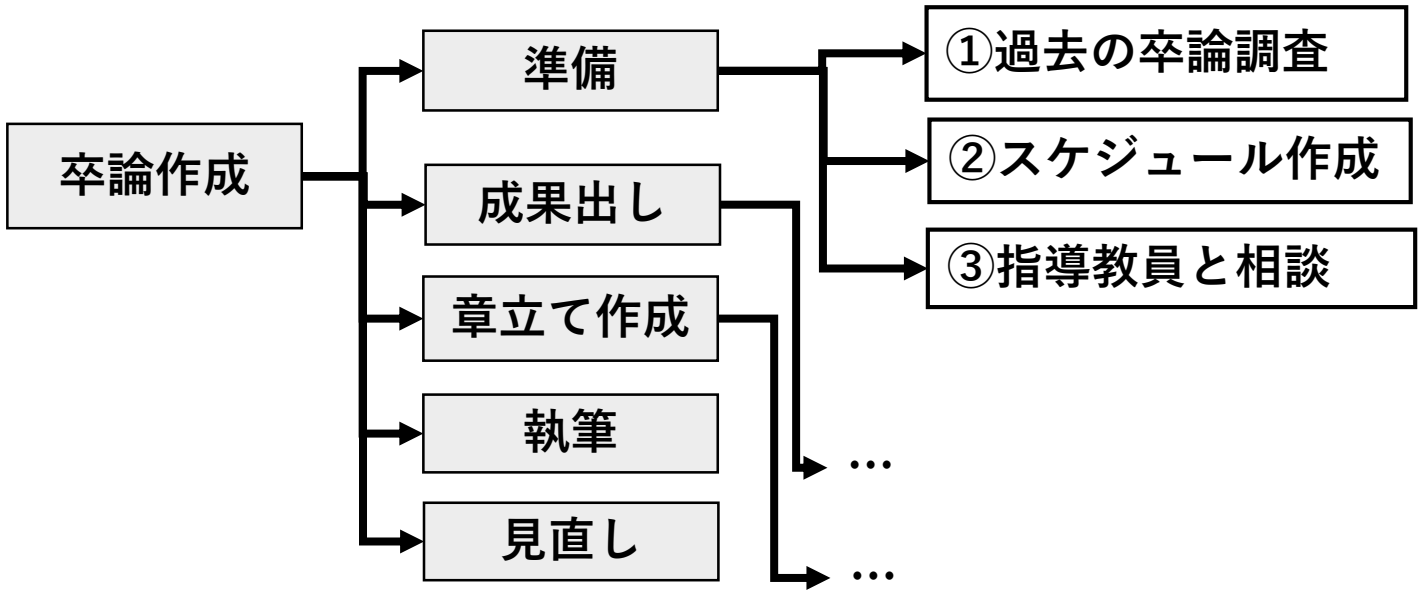
　全体から部分へ。最上位のWBSを定義して、上位から要素分解する。

②成果物ベース、プロセスベースの作業洗い出しを組合せる

　作成方針1:成果物ベース

例:在庫管理システム→サーバ、ネットワーク、プログラム→各構成設計の仕様

　作成方針2:プロセスベース:時間軸(作業の進め方) 例:基本設計、詳細設計



# 課題12-2:締切7/21

1)「グループワークで考案した案」のWBSをまずは個人で作成ください。各作業にかかりそうな時間も記載ください。

**WBSは「案の仕様書作成」でも良いし、「案の開発」でも良いし、「案の教材作成」でも良いです。**

#「数時間～半日以上くらいかかる作業」を最小単位としてください。

#細かすぎる作業は書かなくて良いです。

2)個人で作成したWBSをグループワーク班で共有して、考慮漏れが無いか確認ください。

グループワークですが、下記を各人で提出ください。

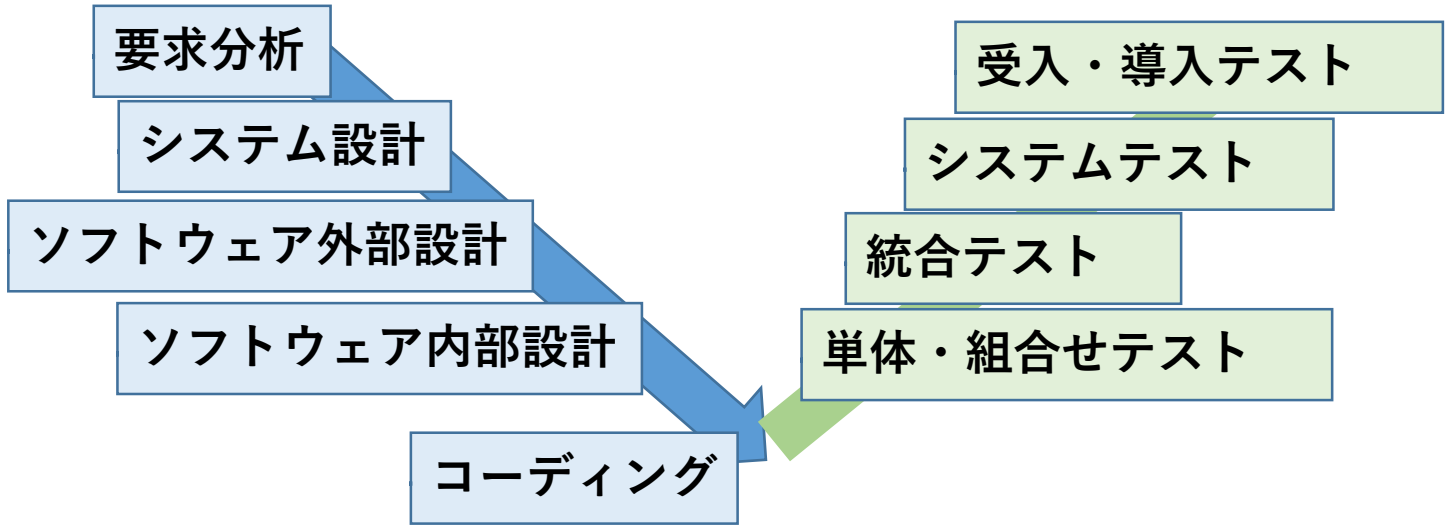
提出1)個人で作成したWBS(考慮漏れなどあれば修正後)を提出ください

2)グループワーク班で共有した自分や相手の感想を提出ください。

# プロセス定義

開発作業を効率的に遂行し、管理するための作業の骨組みを定めることをプロセス定義という。プロセスは、大まかな開発段階(フェーズ)分けと各段階での成果物が設定されている。

通常、各企業は組織内にいくつか標準プロセスを持っており、開発するソフトウェアの性格に応じて仕立てて使用することが多い。例えば、ウォータフォールモデルはよく使われるモデルの一つである。管理プロセス例としては、設計と試験を対応付けたV字型モデルやスパイラルモデルがある。



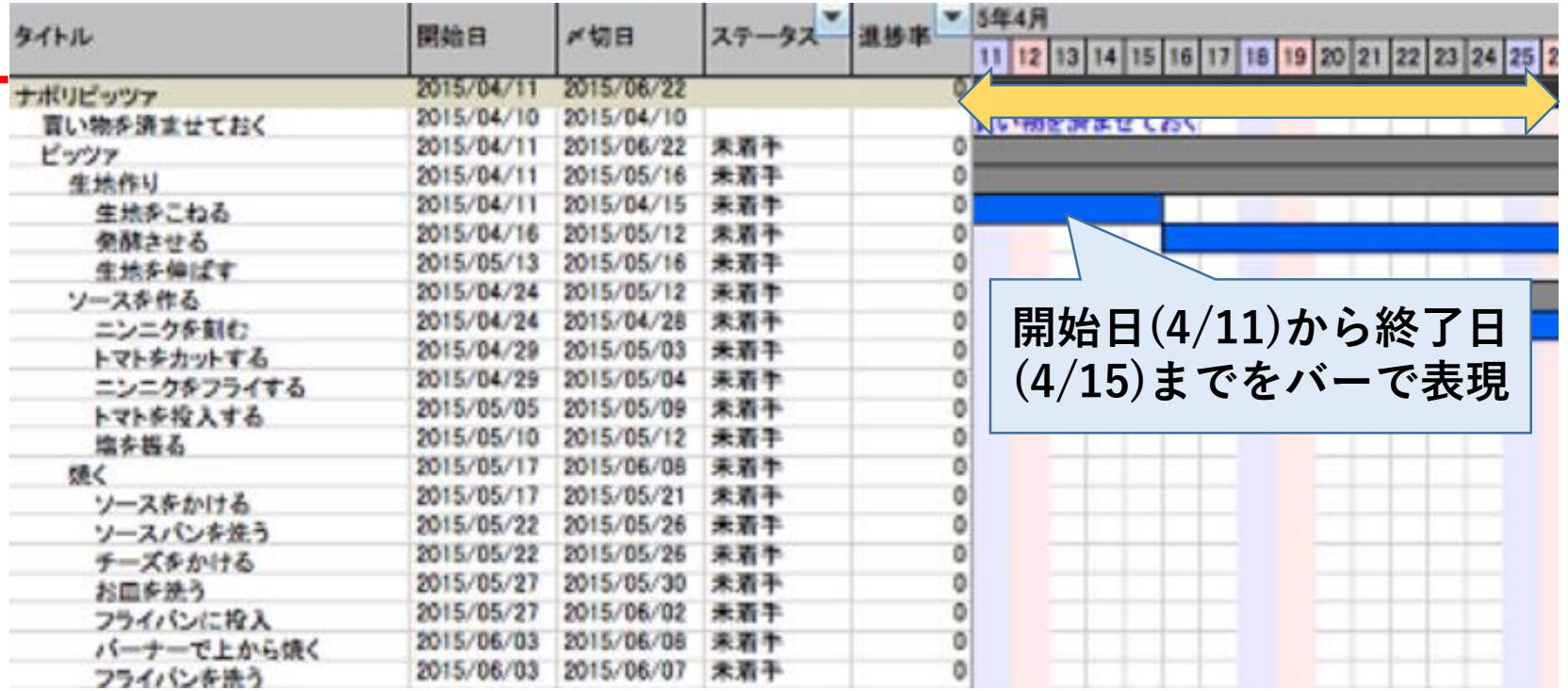
# 進捗管理:スケジュール計画1

**スケジュール計画:**作業リストと各作業の工数見積り、および各作業を担当する要因の関係を記述した責任分担表を入力と、各作業の開始日と終了日を設定する作業のこと。ガントチャートが簡便な手法としてよく使われる。

**ガントチャート:**縦軸に作業項目、横軸にカレンダー

各作業の開始日から終了日までをバーで表現

バーを塗りつぶして作業計画に対する実績分を割合で表現

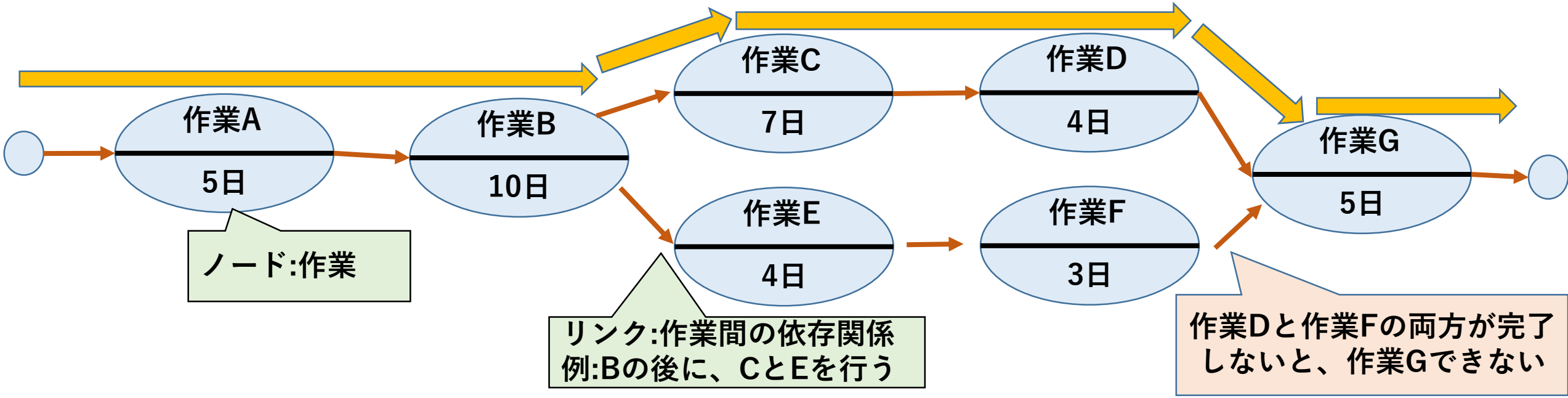



https://image.itmedia.co.jp/l/im/ait/articles/1504/24/l_gantt01_a.png#_ga=2.232284570.468402460.1643333675-1380744308.1611302470

# 進捗管理:スケジュール計画2

**クリティカルパス法:**各作業をノード、作業間の依存関係をリンクとする作業ネットワーク図を構成し、各作業に所要時間を記述することで、終了日や終了日から遡って開始日を計算する数理的な方法

クリティカルパス:最長の作業パスのこと。下記ではA→B→C→D→Gの31日となる。各作業の所要日数が短縮されなければ、31日が最短時間となる。



# (余裕があれば)課題12-3:締切7/21

課題12-2で作成したWBSのワークパッケージについて、作業ネットワーク図を作成し、おおまかな作業日数を記載した後に、クリティカルパスを求めてみてください。