ビースラッシュの
設計力向上セミナー

2020年6月1日

ビースラッシュ株式会社

内容

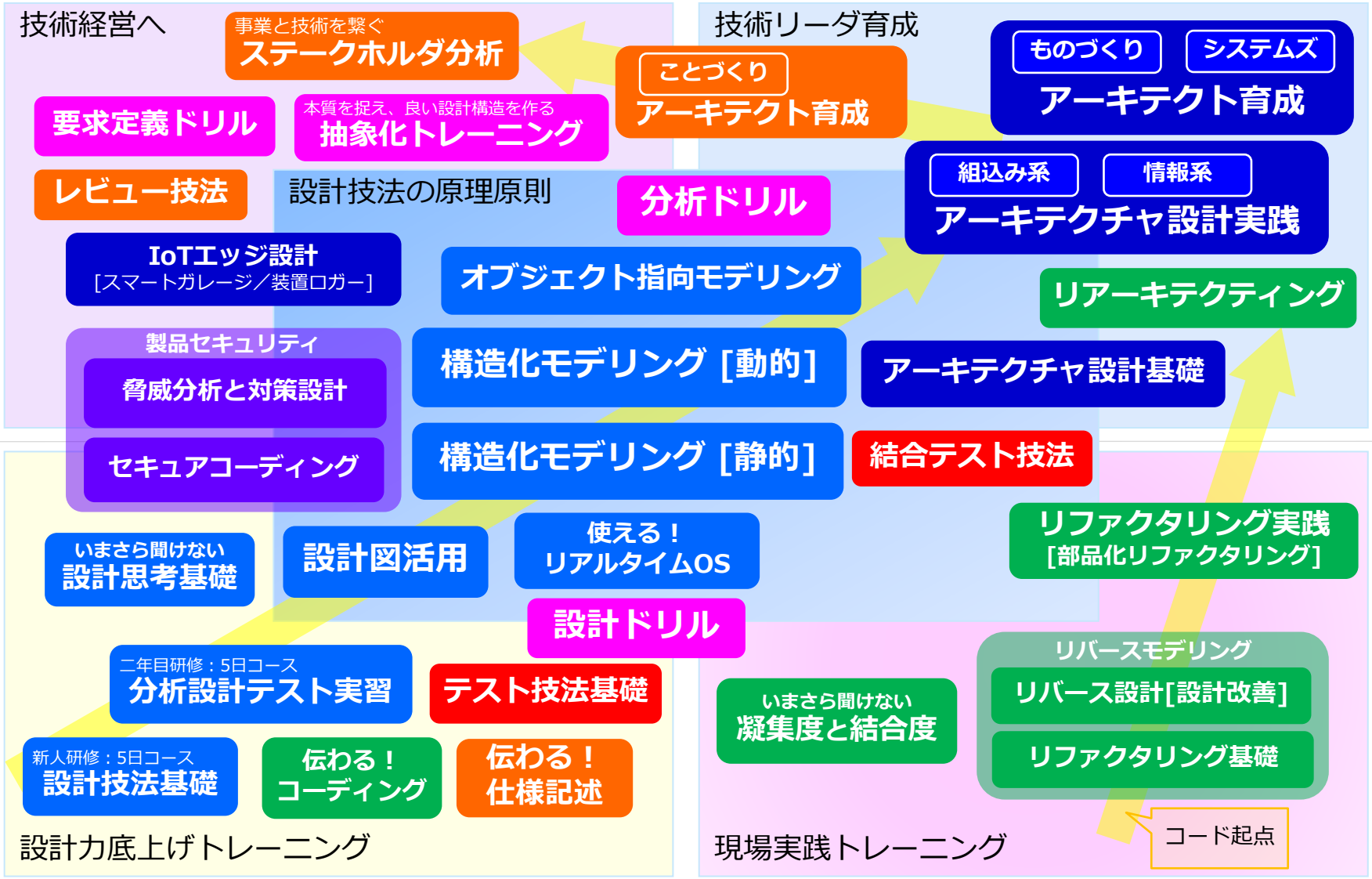
1．設計力セミナー全体マップ

2．アーキテクトまでのスキルパス

3．目的別セミナー紹介

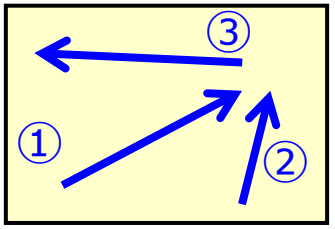
4．開催形態（集合／オンライン／多地点）

1. 設計力向上セミナー全体マップ



設計力向上セミナーの狙いと構成

- ソースコードを追いかけるコード中心開発から、設計図を見る設計中心開発への体質変換を目指しています



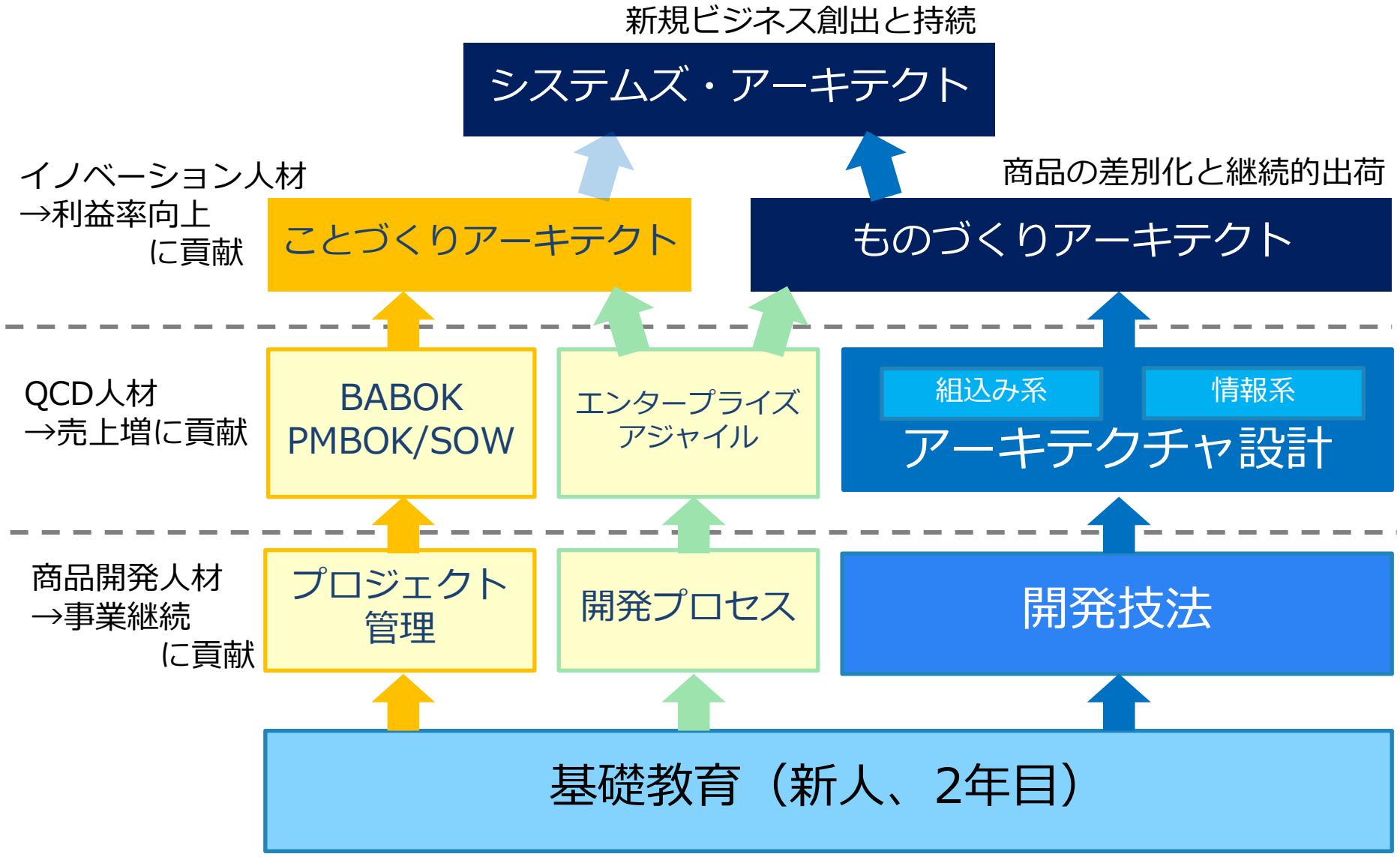
① 左下から右上に向けて設計力向上を計画的に実施する
- 左下：新人／２年目から、まずは設計図を見る習慣づけ
- 中央：経験を少し積んだ時点で、理論とと実践を合わせ込む
- 右上：設計図を用いて、設計意図を伝達できるアーキテクトへと成長

② 右下から右上に、コードを起点として、ソフトウエアの資産価値を向上させる
- 右下：日々のプログラミング作業時に、リバースモデリング（リファクタリングして、コードから設計図を作る）をすることで、徐々に設計中心開発へと変革を実現する
- 右上：ソースコードを起点として、アーキテクチャを図面化し、アーキテクチャを改善しながら、部品化再利用／プロダクトライン開発ができるソフトウェア資産を作る

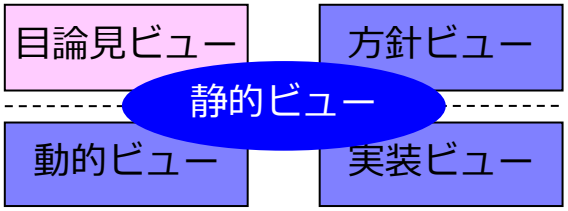
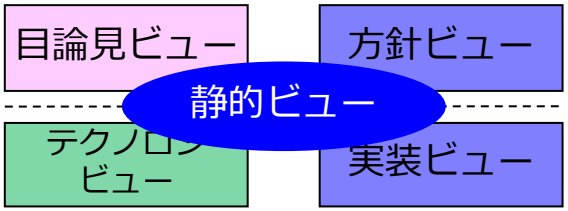
③ 右上から左上で、アーキテクト活動を強化する
- アーキテクトとして、経営とテクノロジをつなぐ為に、次の３つのスキルを強化する
  1．ビジネスを見据える力　⇒ステークホルダ分析
  2．本質を捉える力　⇒抽象化トレーニング
  3．「もの」と「こと」の融合　⇒ことづくりアーキテクト

2. アーキテクトまでのスキルパス

アーキテクトまでのスキルパス

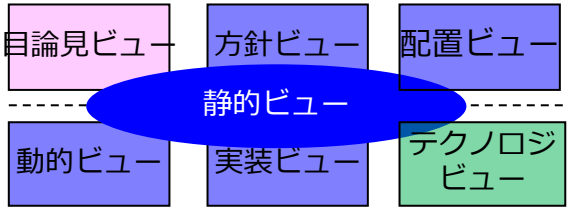
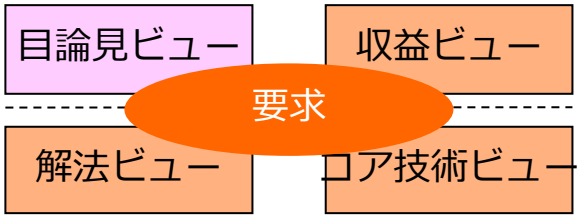


アーキテクチャ設計／アーキテクト育成の概要

コース	概要	要点
ものづくりアーキテクト （3日間コース）	アーキテクチャ設計を活用した技術リーダーシップ。 全体構造（静的構造）を中心に設計意図を明確にして、ビジネスとテクノロジをつなぎます。	アーキテクチャを図表化して、設計意図の周知・調整するリーダーシップ。 ・複数ビューでの図表化と統合 ・アーキテクチャドキュメントの作成 ・技術リーダーシップの役割
アーキテクチャ設計実践 [組込み系] （3日間コース）	組込みシステムのアーキテクチャ構築。 ・静的ビューで全体を俯瞰 ・動的ビューで性能と状態を定義 ・実装ビューでコードの規則性	5つのビューで図表化 
アーキテクチャ設計実践 [情報系] （2日コース）	エンプラ系／Web系システムのアーキテクチャ構築。 ・静的ビューで全体を俯瞰 ・テクノロジビューで技術活用 ・実装ビューでアンチコードを排除	5つのビューで図表化 
アーキテクチャ設計基礎 （1日コース）	アーキテクチャ設計への動機づけ。	設計技法とアーキテクチャ設計の違い。 アーキテクチャ設計における典型的なビューとその図面の例。

2つのオプション

- IoTシステムに対応する「システムズ・アーキテクト」
- 要求定義中心の「ことづくりアーキテクト」

コース	概要	要点
システムズ・アーキテクト （3日間コース）	異なるドメインのモジュールをつなぎ、ひとつのシステムを構築します。 配置ビューで、複数パートナーとの連携と各インタフェースを定義します。	７つのビューで図表化（TBD） 
ことづくりアーキテクト （3日間コース）	複数ステークホルダ／複数システム（システムオブシステムズ）に対応できる技術リーダーシップ。 要求を中心に複数ビューをつなぎます。	5つのビューで図表化 

アーキテクトの基本スキル

3つの基本スキル

①複数のビューポイントで対象を図表化する

- ビジネスとテクノロジを繋ぐ
- 異なるビューポイントに補助線を引く
- 両方の視点で見ることができないとアーキテクトにはなれない

②複数の図表を統合して一冊のアーキテクチャドキュメントにする

- 抽象化と統合化のスキル
- 全体を俯瞰して、かつ、際どい箇所を押さえる

③様々なステークホルダを巻き込んで、リーダーシップを発揮する

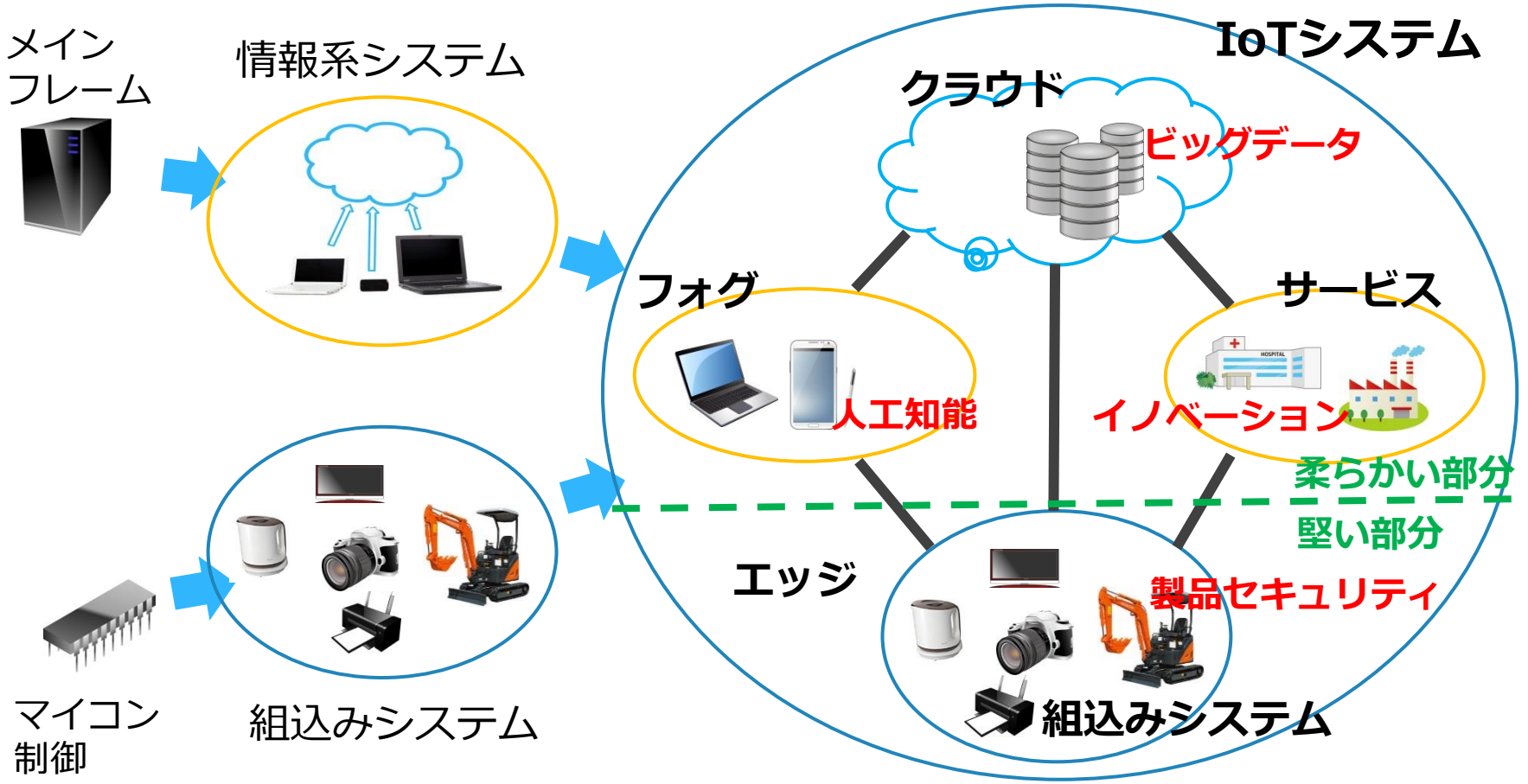
- あらゆる場で判断を行う
- コミュニケーションとリーダーシップのスキル
- アーキテクチャドキュメントで意図を伝え、実施してもらう

アーキテクトの効用

視点	局面	要点
開発速度	開発作業	無駄なことをしない。手戻りがない。
	複数人開発	設計方針の共有。効率的なすり合わせ。
	遠隔地開発	コミュニケーションギャップの低減。 オフショアでの異言語・異文化の理解。
納期	エンジニアからの報告	予実績の差異。定量的報告。プロアクティブ。
	マネジャからの指示	場当たり的な指示をしない。丸投げしない。
	工数見積り	属人的な勘と度胸だけに頼らない。
	変更管理	やってみないと分からない、からの脱出。
品質	品質の作り込み	上流工程での品質の作り込み。
	劣化防止	一時しのぎのパッチやジャンパー線の管理。
	再発防止	現象ではなく、原因を探り、設計カイゼンへ。
育成	技術伝承	OJT。一子相伝から複数人育成へ。
	設計意図の周知	チーム全体のベクトル合わせ。

参考：IoTシステムとは

- 近年、ネットワークに接続する製品の開発が増加
- IoTでは、製品を取り巻く周辺の状況変化が発生



参考：IoTとシステムズ・アーキテクト

求められること

(1)異ドメインの接続　　（ヘテロジニアスな全体像をつかむ）
(2)ビジネスへの補助線　（コンセプトやバリューを考える）
(3)本質を見抜くスキル　（抽象化できる人材の育成）

JEITAワークショップ2016より、山田まとめ
http://home.jeita.or.jp/cgi-bin/page/detail.cgi?n=933&ca=1

3つの基本スキル

①複数のビューポイントで対象を図表化する
- ビジネスとテクノロジを繋ぐ
- 異なるビューポイントに補助線を引く
- 両方の視点で見ることができないとアーキテクトにはなれない

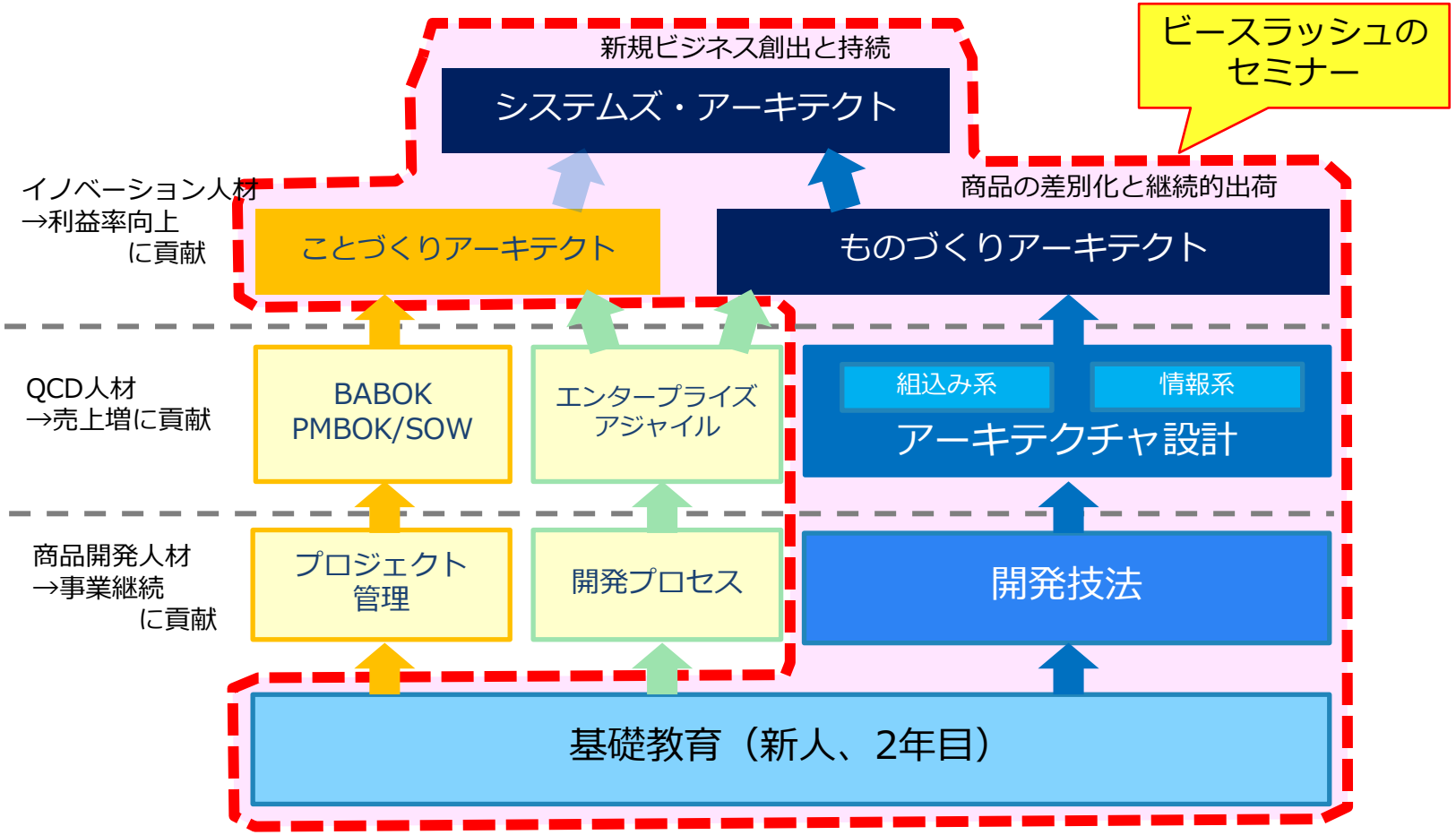
②複数の図表を統合して一冊のアーキテクチャドキュメントにする
- 抽象化と統合化のスキル
- 全体を俯瞰して、かつ、際どい箇所を押さえる

③様々なステークホルダを巻き込んで、リーダーシップを発揮する
- あらゆる場で判断を行う
- コミュニケーションとリーダーシップのスキル
- アーキテクチャドキュメントで意図を伝え、実施してもらう

スキルパス別セミナー分類

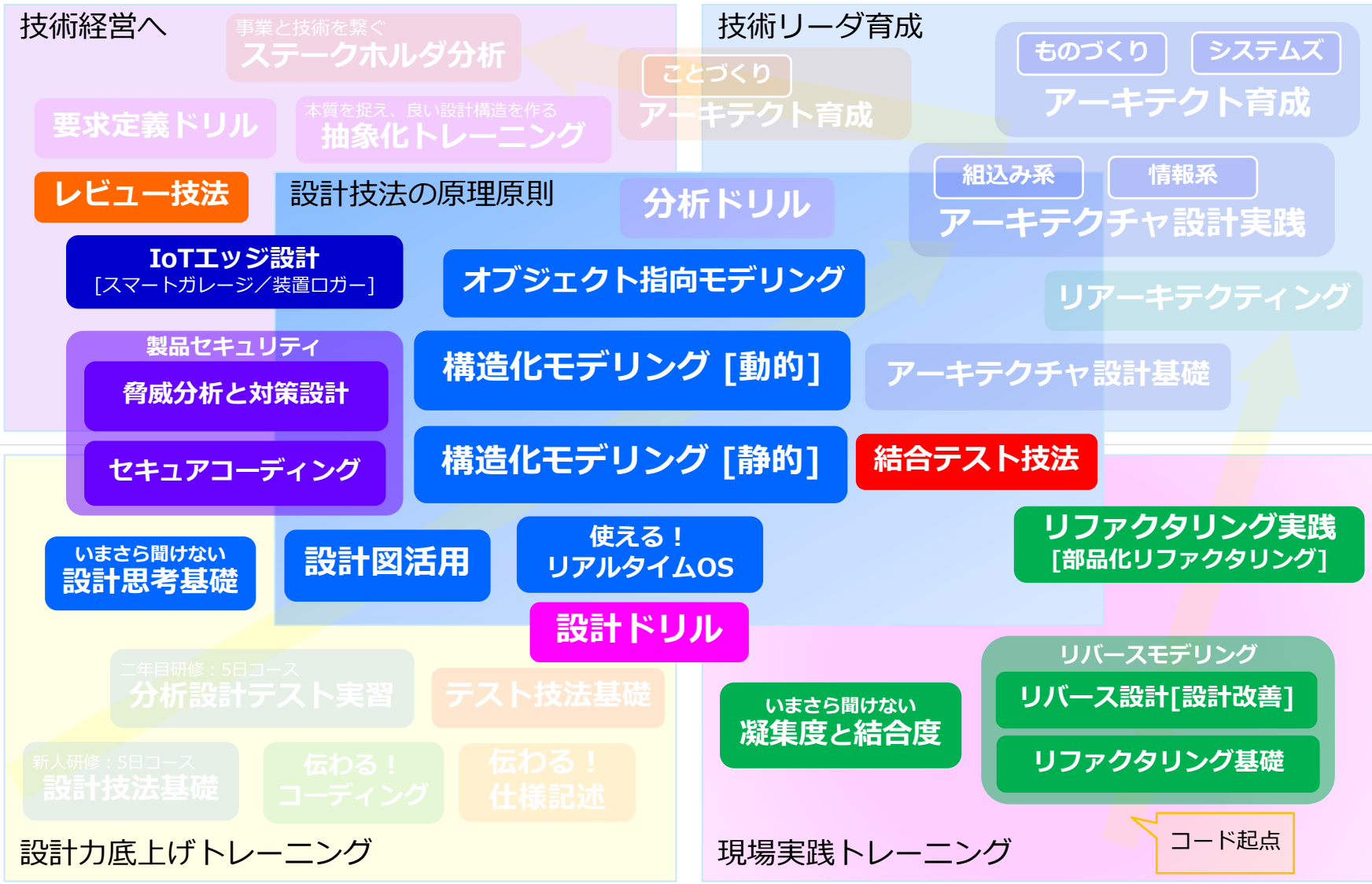
ビースラッシュのセミナー

- 設計エンジニアリング系のセミナーを提供
  - プロジェクト管理／開発プロセスは、提供していません



基礎教育

開発技法



アーキテクチャ設計

アーキテクト

3. 目的別セミナーの紹介

目的別セミナーの狙いと主な対象者

	目的別セミナー	狙い	主な対象者
A	設計力底上げ	設計図を活用したソフトウェア開発ができる	新入社員、2年目社員 基本から学びたい経験者
B	理論に基づく実装	得意分野を作り、自信をもって開発に取り組むことができる	新入社員、2年目社員
C	体質改善 コード中心から設計中心へ	自己流の開発スタイルから脱却し、ソースコードよりも先に設計図を見る習慣ができる	2年以上の開発経験者
D	レガシーコード資産化	既存コードを設計改善して、ソフトウェア資産を活用した開発に移行することができる	2年以上の開発経験者
E	アーキテクト	全体を俯瞰し、際どい個所を明確にして、開発をリードすることができる	3年以上の開発経験者 アーキテクトを目指す方
F	システムズアーキテクト	異分野のシステムを統合することで、価値あるシステムを作り上げることができる	3年以上の開発経験者 アーキテクトを目指す方
G	品質力底上げ	ソフトウェア品質作りのための技法を使うことができる	2年以上の開発経験者 及び、テスト技術者
H	実力診断	ドリル形式で解いていくことで、自分の実力と弱点を把握することができる	2年以上の開発経験者
I	製品セキュリティ	製品設計とソースコードに含まれる脆弱性を理解し、堅牢な製品開発を推進できる	3年以上の開発経験者 セキュリティエンジニア

目的別のお薦めセミナー

		1	2	3	4	5
A	設計力底上げ	設計技法基礎	分析設計テスト実習	構造化モデリング[静的]	構造化モデリング[動的]	オブジェクト指向モデリング
B	理論に基づく実装	伝わる！ コーディング	伝わる！ 仕様記述	テスト技法基礎	使える！リアルタイムOS	
C	体質改善 コード中心から設計中心へ	設計思考基礎	設計図活用	凝集度と結合度	リバースモデリング	
D	レガシーコード資産化	リファクタリング実践	アーキテクチャ設計基礎	リアーキテクティング		
E	アーキテクト	アーキテクチャ設計基礎	アーキテクチャ設計実践	ものづくりアーキテクト		
F	システムズアーキテクト	アーキテクチャ設計基礎	IoTエッジ設計	ことづくりアーキテクト	システムズアーキテクト	ステークホルダ分析
G	品質力底上げ	テスト技法基礎	結合テスト技法	レビュー技法		
H	実力診断	設計ドリル	分析ドリル	要求定義ドリル	抽象化トレーニング	
I	製品セキュリティ	セキュアコーディング	脅威分析と対策設計			

組込み系と情報系の両方に対応

- ソフトウェア工学に基づいているので設計全般に有効です
  - 組込み系を意識した構成にはなっていますが、
  - 情報系（エンプラ系／Web系）の方も受講できます

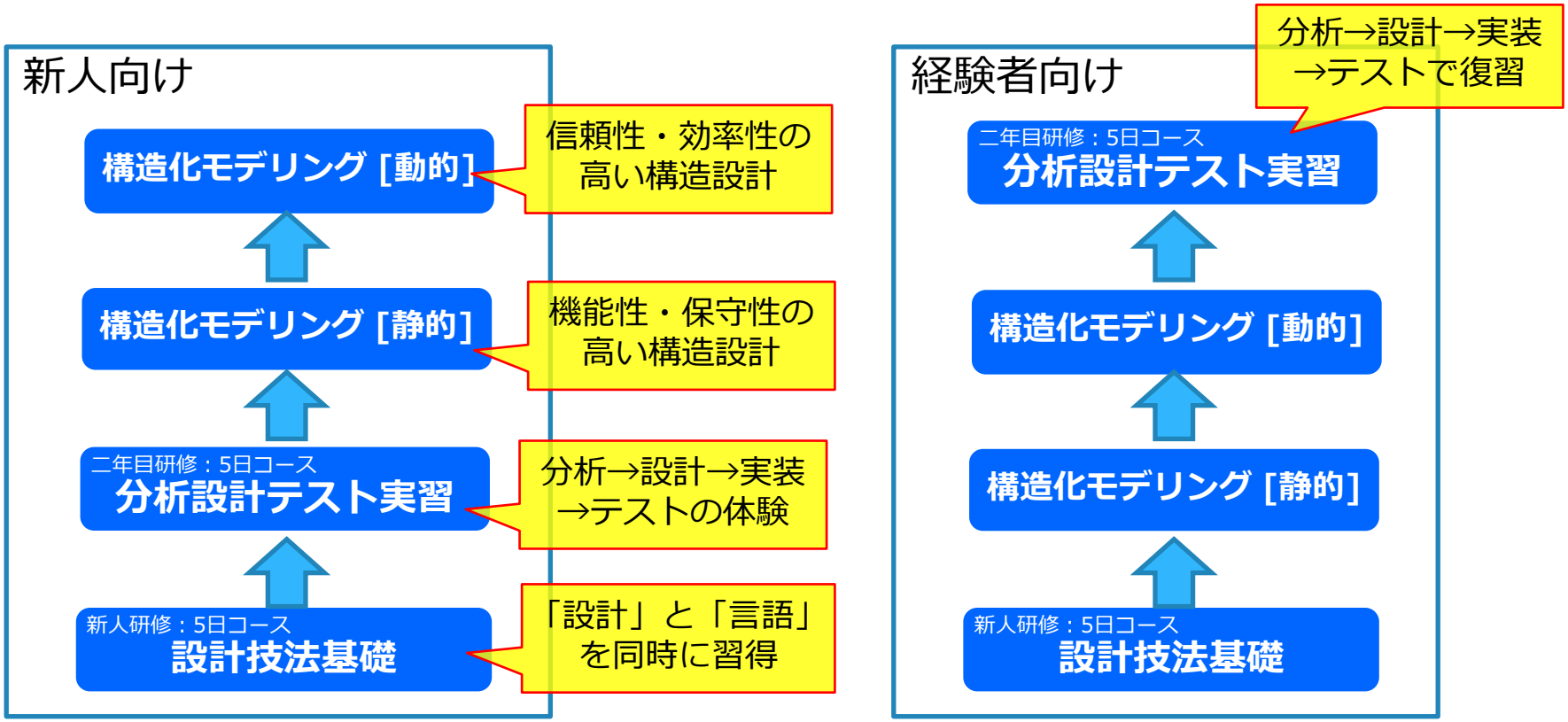
- 情報系の方は、以下の項目をご了承ください
  - プログラム言語はC言語が中心です
  - 動的ビューに言及しています
    - 情報系ではあまり意識しないタスク設計が含まれます
  - 演習として、ハードウェア基板を使っているセミナーがあります
  - IoT時代では、エッジ側（HW側）も理解できると、さらに強みになります

A.設計力底上げ

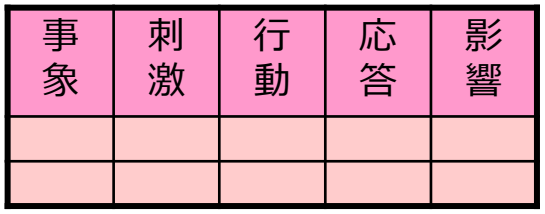
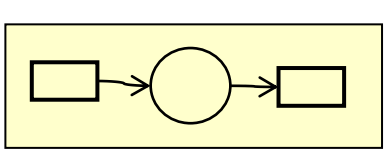
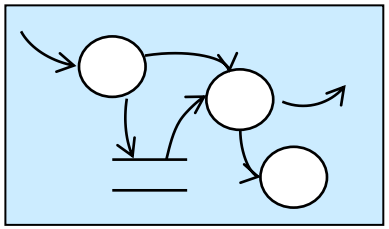
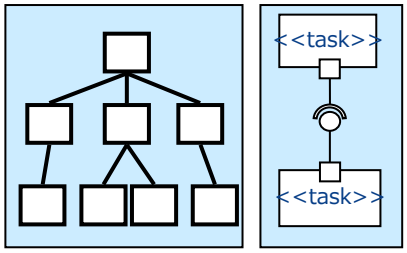
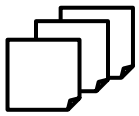
A. 設計力底上げ

A. 設計力底上げ

- 新人の方：経験を積みながら、理論を習得する
  - 事前に「C言語プログラミング」研修を受講してあるとより効果的です
- 2年目以降の方：理論を習得して、それを実践で活用する
  - オブジェクト指向開発の方は「オブジェクト指向モデリング」の受講もお薦めします



各セミナーの適用範囲

工程	主な成果物	モデル例	視点
要求 モデリング	イベントリスト タイミング仕様書		Value
分析 モデリング	コンテキスト図		Scope
	データフロー図		What
設計 モデリング	関数構造図[静] タスク構造図[動] 状態遷移図		How
実装	ソースコード		How

構造化モデリング［動的］

構造化モデリング［静的］

分析設計テスト実習

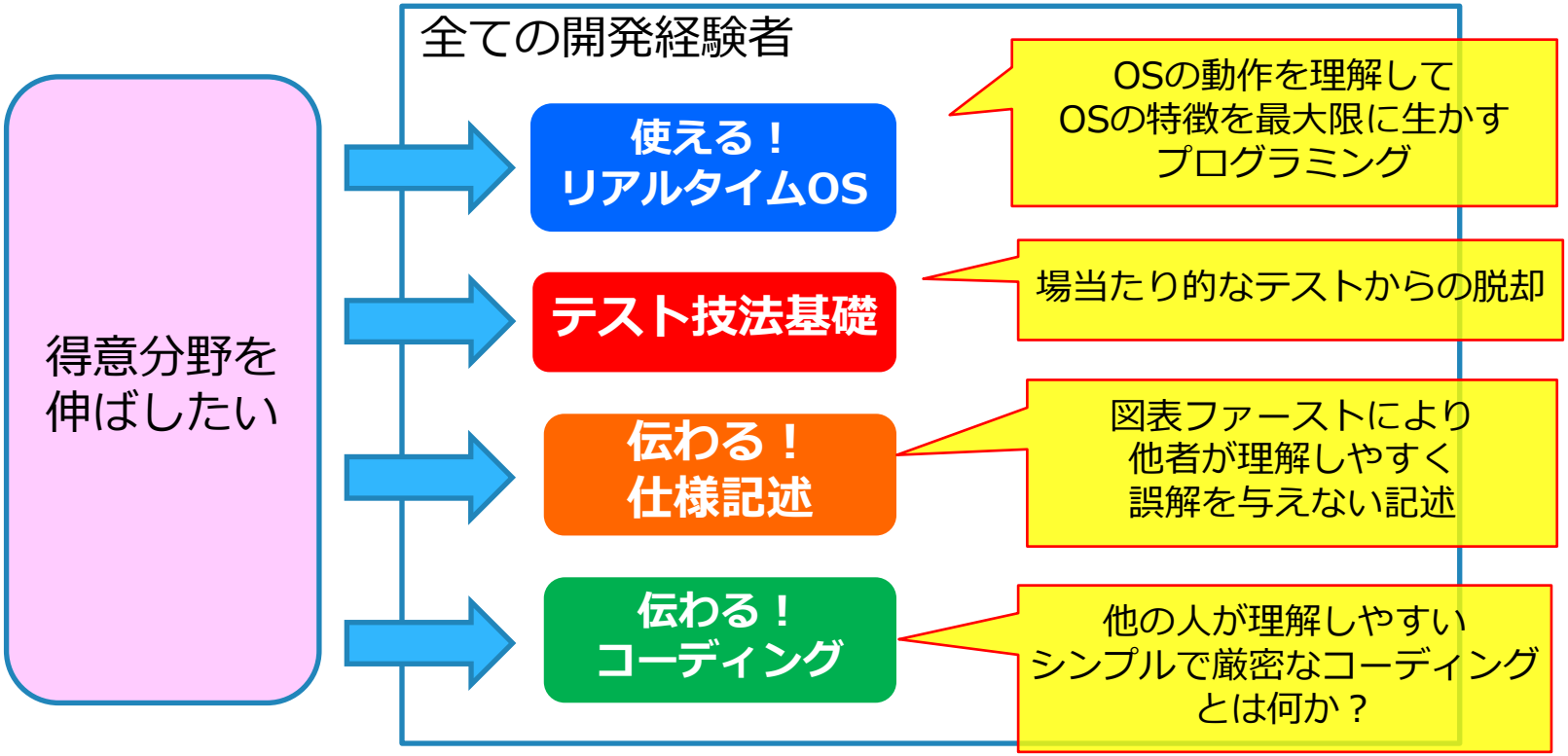
設計技法基礎

B. 理論に基づく実装

B. 理論に基づく実装

B. 理論に基づく実装

- 基礎となる「理論」と「実践」を結び付けて学習します
  - 理論を習得することで、確実なスキル獲得ができます
  - 自分の強みを明確にすることで、理論に裏付けされた実力となります



主な内容

使える！リアルタイムOS

- 時間制約の設計
- タスクとはタスクの状態／優先度
- タスク間通信
- データ共有
- 静的構造と動的構造
- リアルタイムOSを使う最大の理由

テスト技法基礎

- 有効同値クラスと無効同値クラス
- 境界値分析
- 決定表
- エラー推測
- テストカバレッジ
- テストしやすいコード

伝わる！コーディング

- 簡潔さ、読みやすさの基本作法
- ファイル／関数／データの命名
- 公開と隠ぺい
- 演算、分岐／繰返しの表記
- 関数呼び出しと構造
- コーディング規約

伝わる！仕様記述

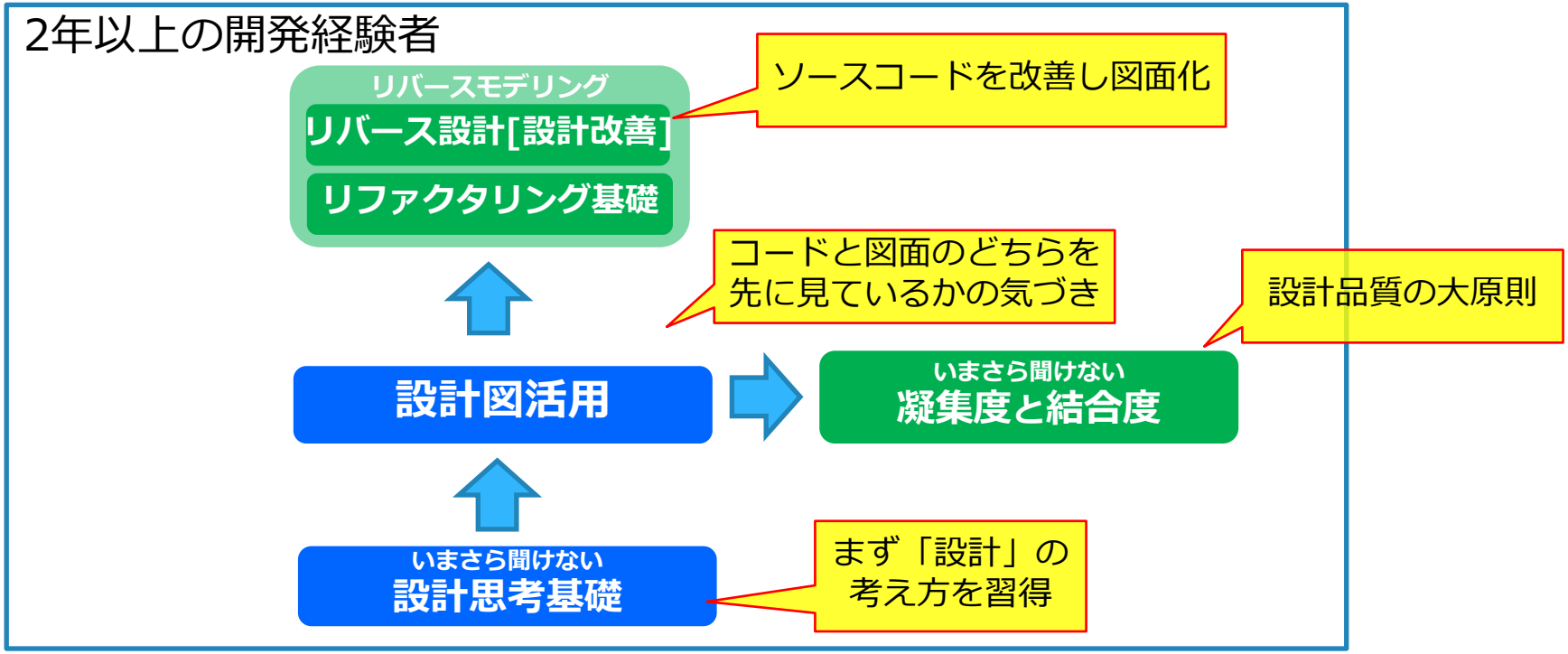
- 「伝わる」とは
- 図表ファースト
- 画面から仕様を読み取る
- 文章から仕様を読み取る
- 会話から仕様を読み取る
- 図表でシンプル＆厳密に表現する

C. 体質変換

C. 体質変換

C. 体質変換

- 「まずコードを見る」から「まず設計図を見る」への開発スタイル変革
  - 「コードを見て、現象を追いかけて、検索で影響範囲を探す」という開発は時間がかかり、かつスキルも上がっていきません
  - 「まず設計図を見て、あたりを付けて、ソースコードで確認する」という開発をすることができます



コード中心から設計中心へ

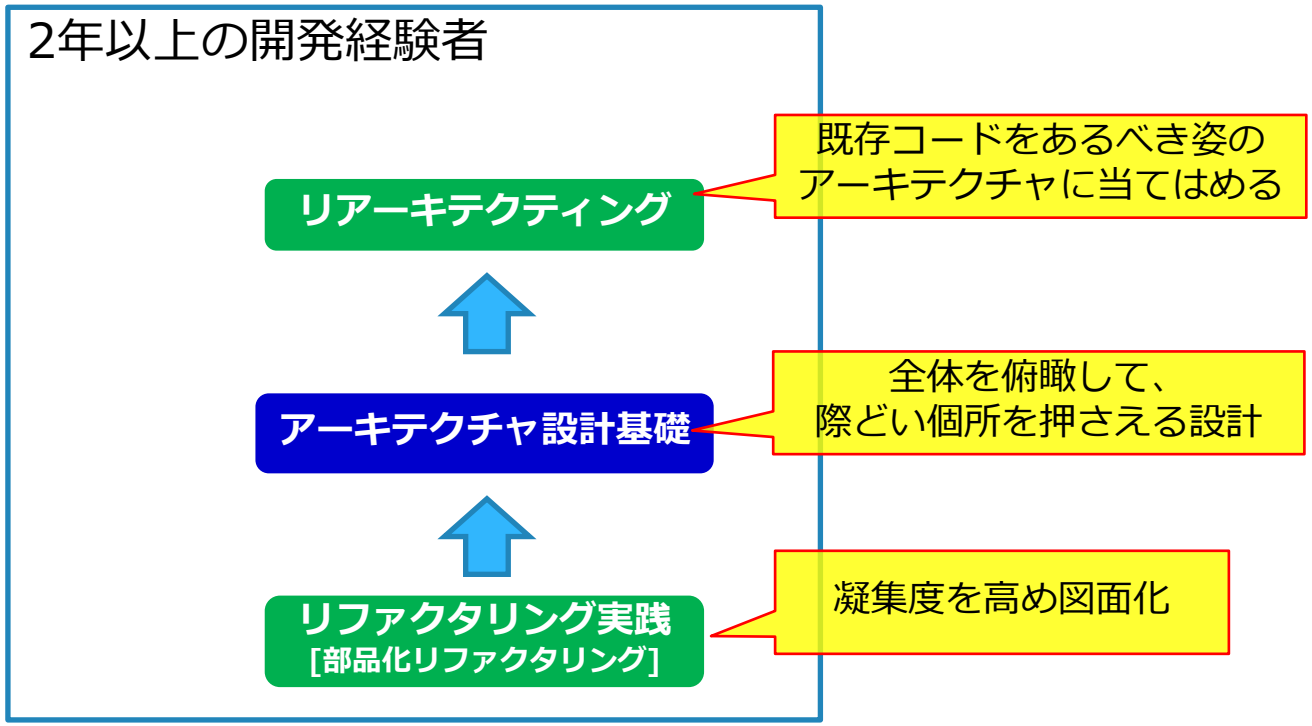
	コード中心 ➡ 設計中心		解説
大規模開発	掛け算	足し算	コード中心では、最初から”すり合わせ”が必要となり、工数は倍倍ゲームとなる。設計中心では、入れ物があらかじめ決まっており、その中を足し合わせて、必要な部分だけ”すり合わせ”る。従って無駄な工数が削減できる。
マネジメント	人	モジュール	コード中心では、担当者にしか分からない属人的な資産であり、その人を管理することになる。 設計があれば、モジュールでの管理が可能。
再利用	アドホック	計画的	出来上がったコードを、次に活用するというアドホックな方法では、品質が徐々に劣化してしまう。あらかじめ再利用できる設計をしておくことが求められる。
外部委託	丸投げ	戦略的	コード中心では、要求を丸投げするか、現場に常駐するかの、いずれかの方法となってしまう。技術流出と技術力の低下も招く危険性がある。設計中心では、資産を特定し、どこを外部委託するかを見極めることができる。
労働形態	労働集約	知識集約	現状のソフトウェア開発は時間対価の労働であり、職種としての魅力も低下してきている。工学系（情報系）離れを防止するためにも、成果物の価値対価の産業へと変革が求められる。
経営課題	人員不足	スキル不足	人員不足の現場では、未経験者の投入が、悪循環を招いている。量から質への転換が必要であり、長期的視野で設計できる人材の育成に取り組まなければならない。

D. レガシーコード資産化

D. レガシーコード資産化

D. レガシーコード資産化

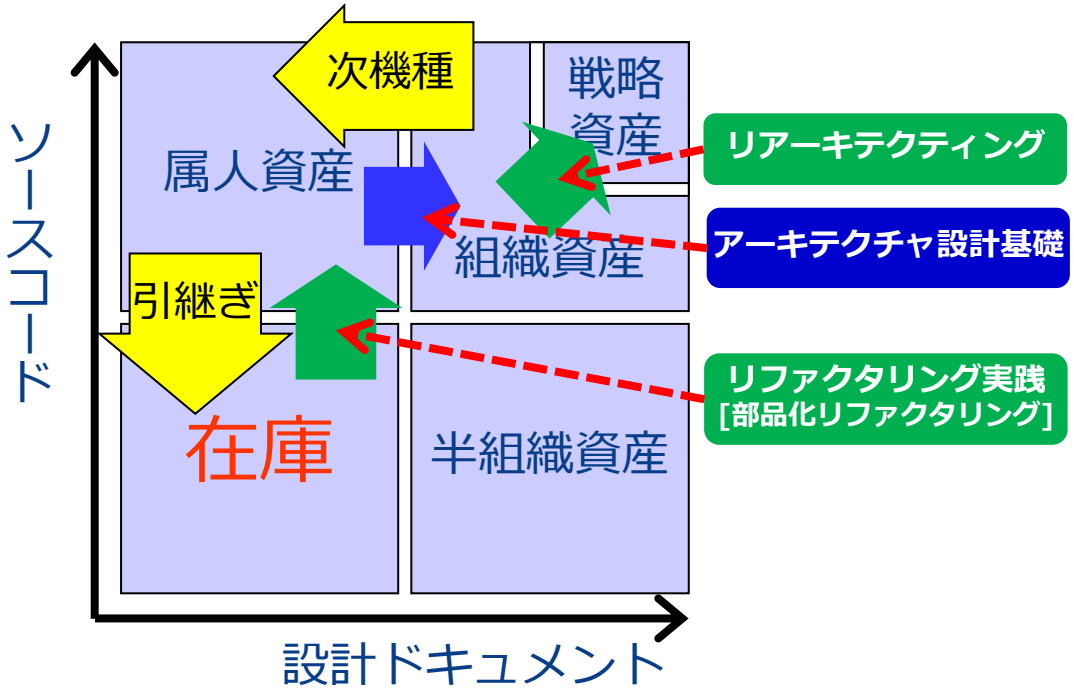
- 今のソースコードの資産価値を高めます
  - リファクタリングにより凝集度を高め、そして、図面化します
  - 複数ビューで図面化することで、アーキテクチャ設計をします
  - 既存コードをアーキテクチャに当てはめて、設計レベルの改善を行います



在庫化を未然防止

資産価値が徐々に低下して『在庫化』していませんか？

- 組織資産⇒属人資産：最初はドキュメントがあったが、誰も読まなくなり、保守されなくなる
- 属人資産⇒在庫：最初は設計ができていたのだが、人が変わったり、仕様追加で劣化する



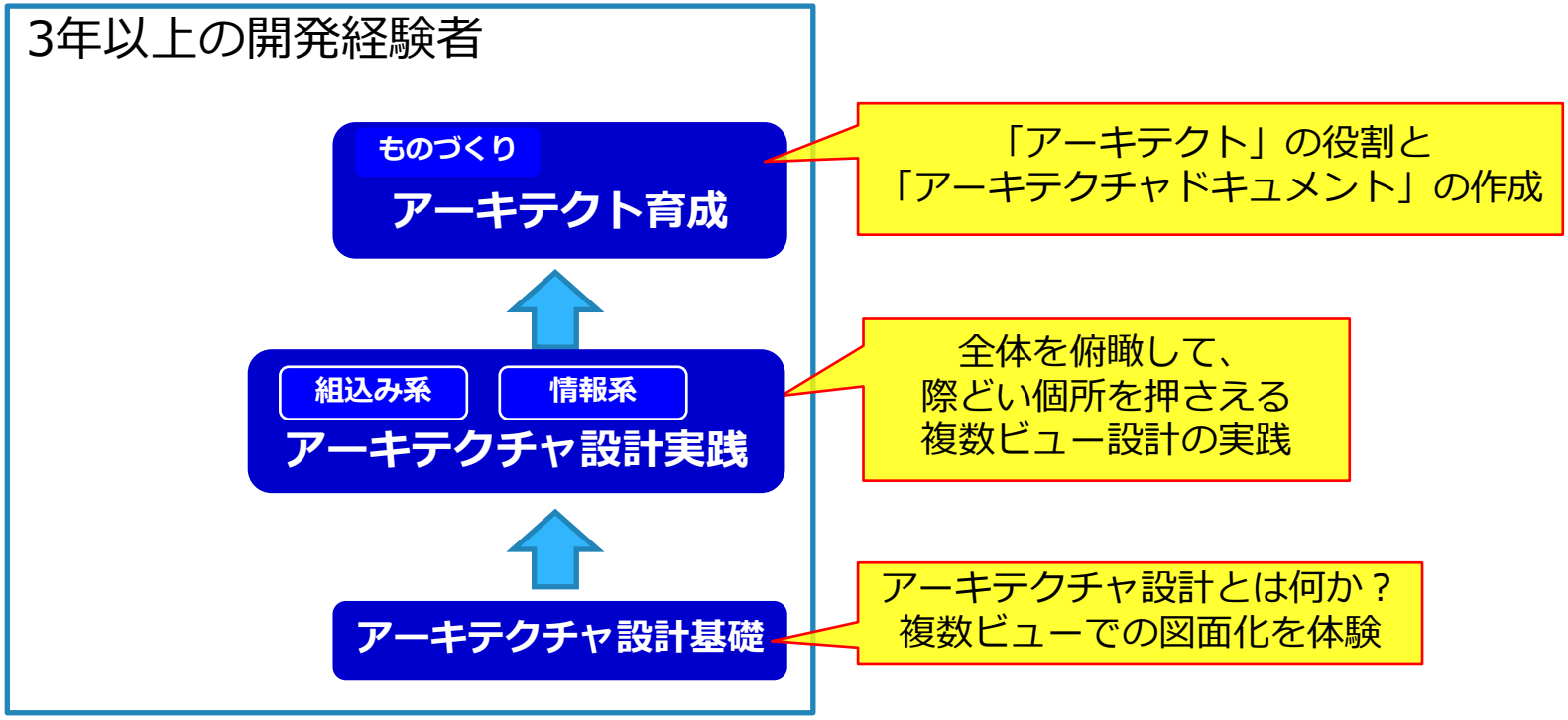
資産名称	特徴
在庫	ソースコードは複雑で説明困難であり、信頼できる設計ドキュメントは存在していない状態。保守コストが膨れていく傾向にある
属人資産	ソースコードはシンプルで分かりやすい。但し設計ドキュメントが揃っていない状態。個人主導の開発であり、引継ぎに苦闘する。
半組織資産	設計ドキュメントが揃っているが、ソースコードが複雑な状態。マネジメント主導の開発である
組織資産	ソースコードがシンプルで、設計ドキュメントが整備されている状態。品質・生産性とも予測可能となる。
戦略資産	ソースコードと設計ドキュメントが統合。戦略的な資産活用ができる。

E. アーキテクト

E. アーキテクト

E. アーキテクト

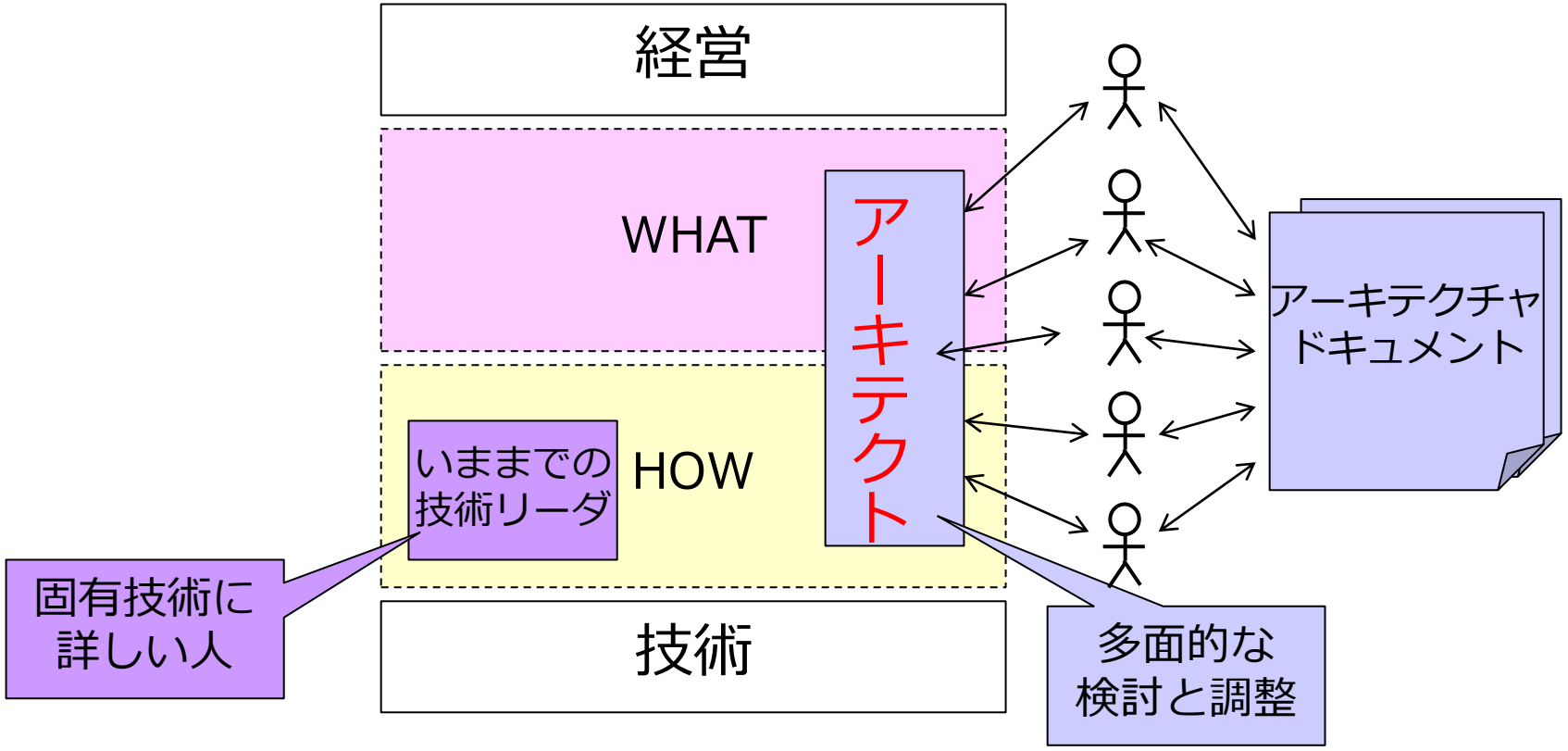
- ビジネスとテクノロジをつなぎ製品開発を成功へ導く「アーキテクト」
- 複数ビューで図面化する「アーキテクチャ設計」
- 設計意図を伝達するための「アーキテクチャドキュメント」



アーキテクトとは

- 『何を、どのように、作るのか』のリーダーシップを発揮する人
  - 設計の方針を決めて、構造設計を行い、
  - 様々な利害関係者と調整し、
  - 価値のある製品開発を推進する

技術と経営をつなぐ



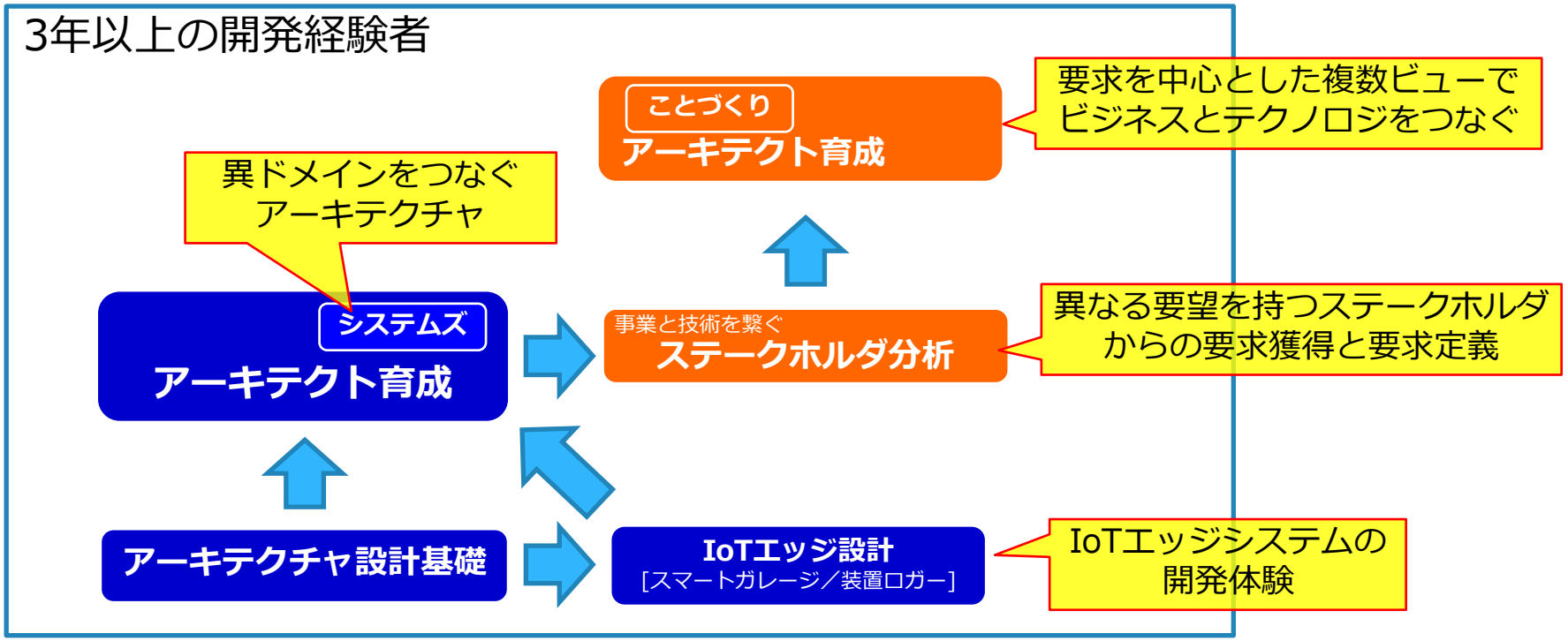
F. システムズアーキテクト

F. システムズアーキテクト

F. システムズアーキテクト

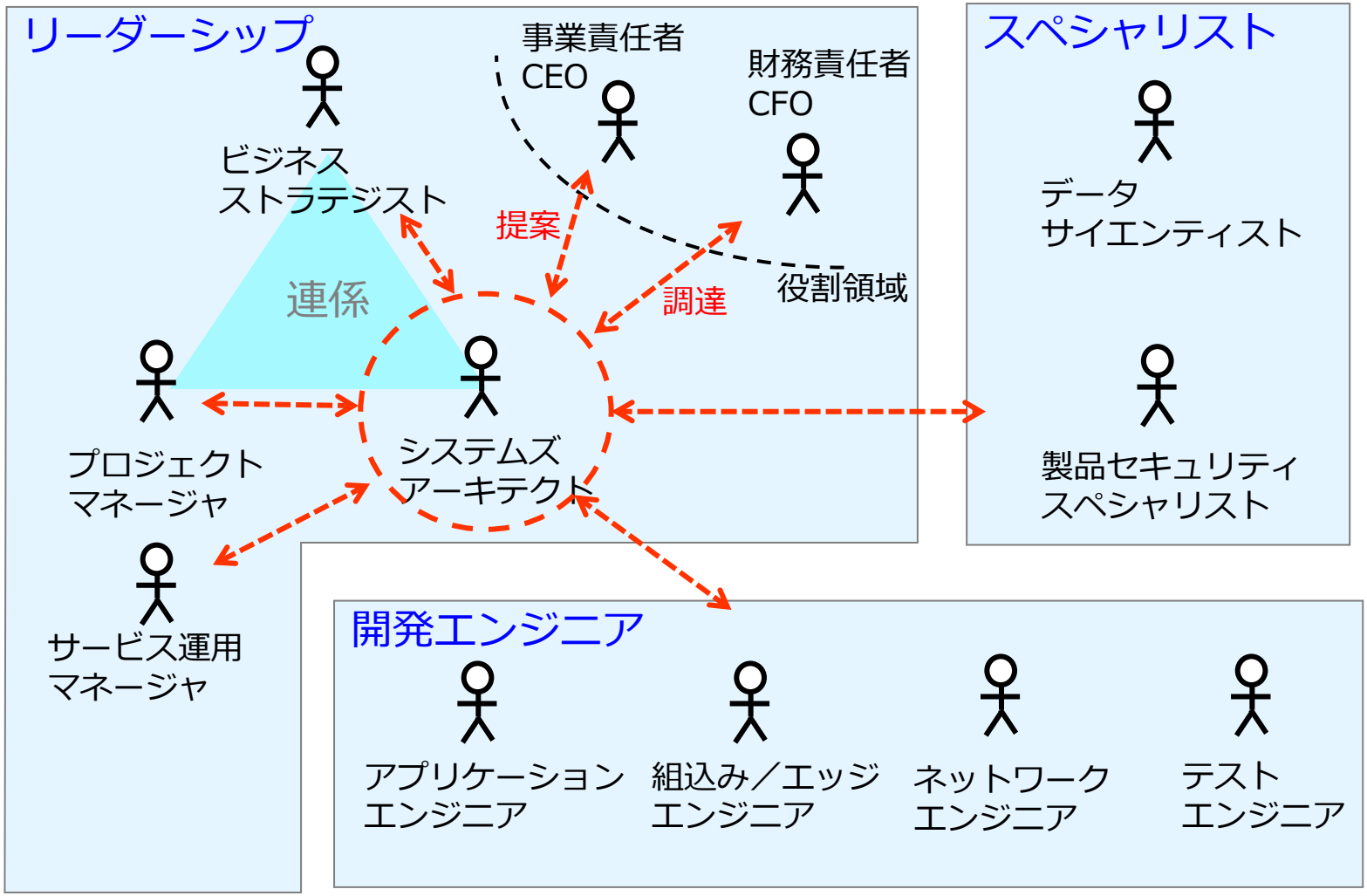
- 複数システムで構成されるIoTシステムのアーキテクチャ設計
  - IoTエッジシステムの開発を経験する

- 複数ステークホルダの要望を要求としてまとめる
- 要求を中心としたアーキテクチャドキュメントでビジネスとテクノロジをつなぐ



システムズ・アーキテクトの立ち位置

- アーキテクトが経営とテクノロジをつなぐ

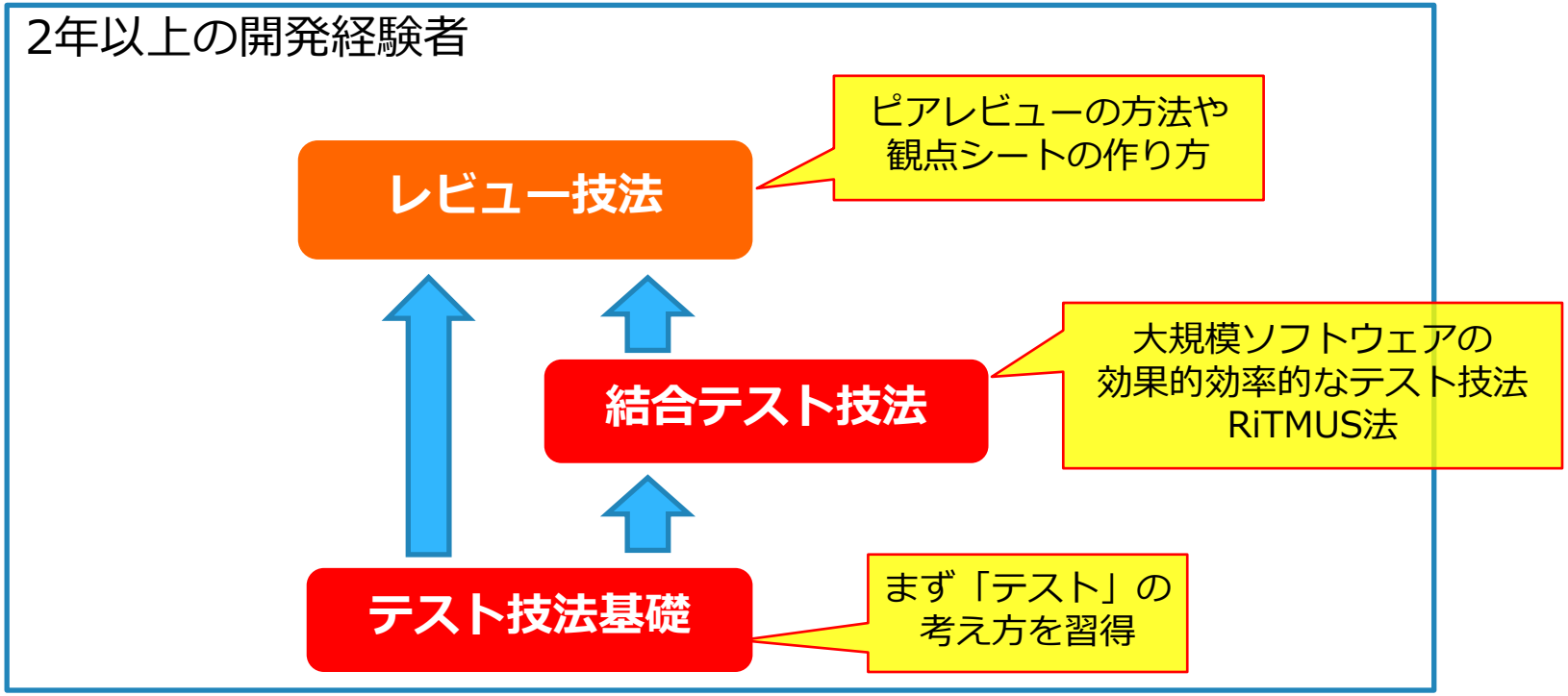


G. 品質力底上げ

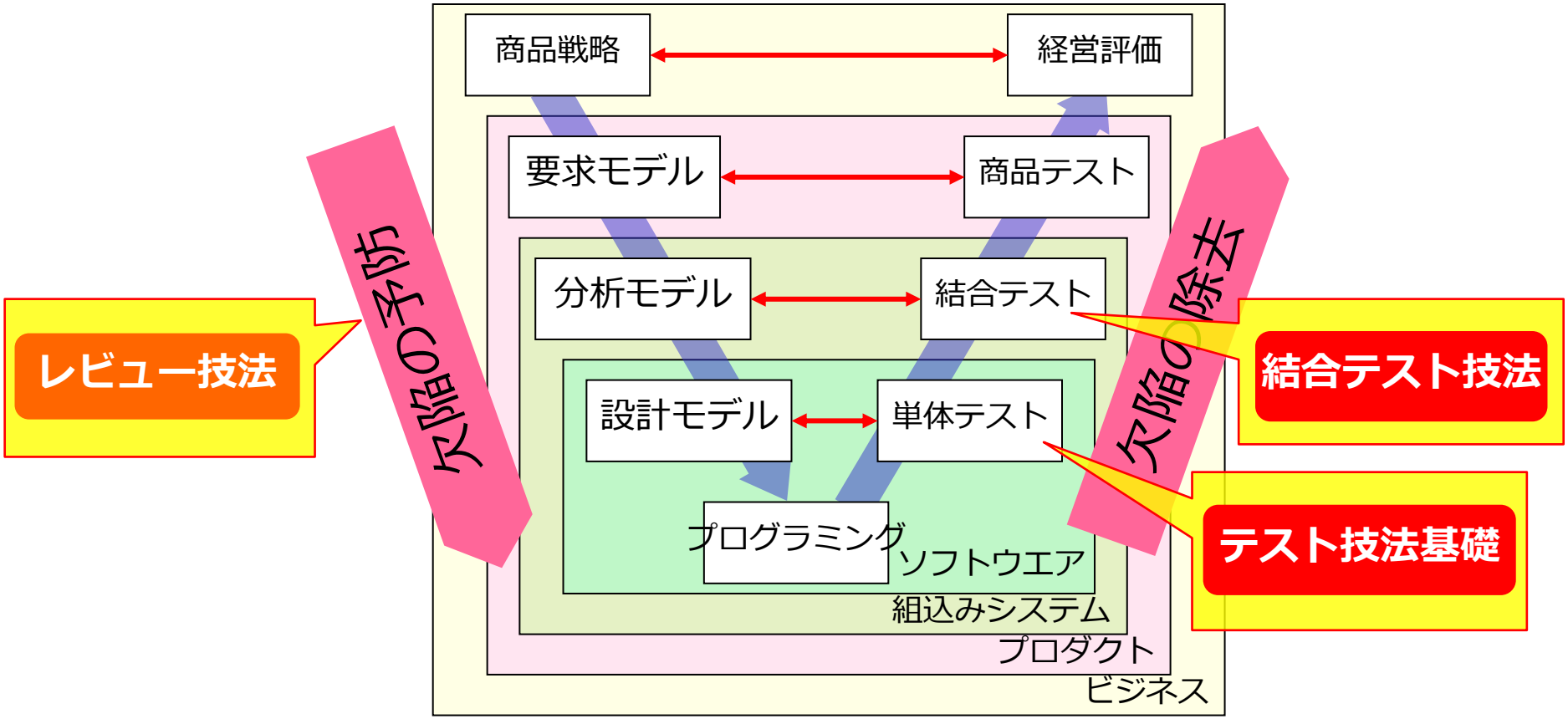
G. 品質力底上げ

G. 品質力底上げ

- テストとレビューによる品質の作り込みを習得します
  - シンプルな設計構造とソースコードは、テストし易い、ことを理解したうえで、
  - テストで、設計時に想定していなかった境界値や組み合わせの不具合を見つける（テスト技法）
  - レビューで、設計成果物を査読技法し不具合の混入を未然防止する（レビュー技法）



各セミナーの適用範囲

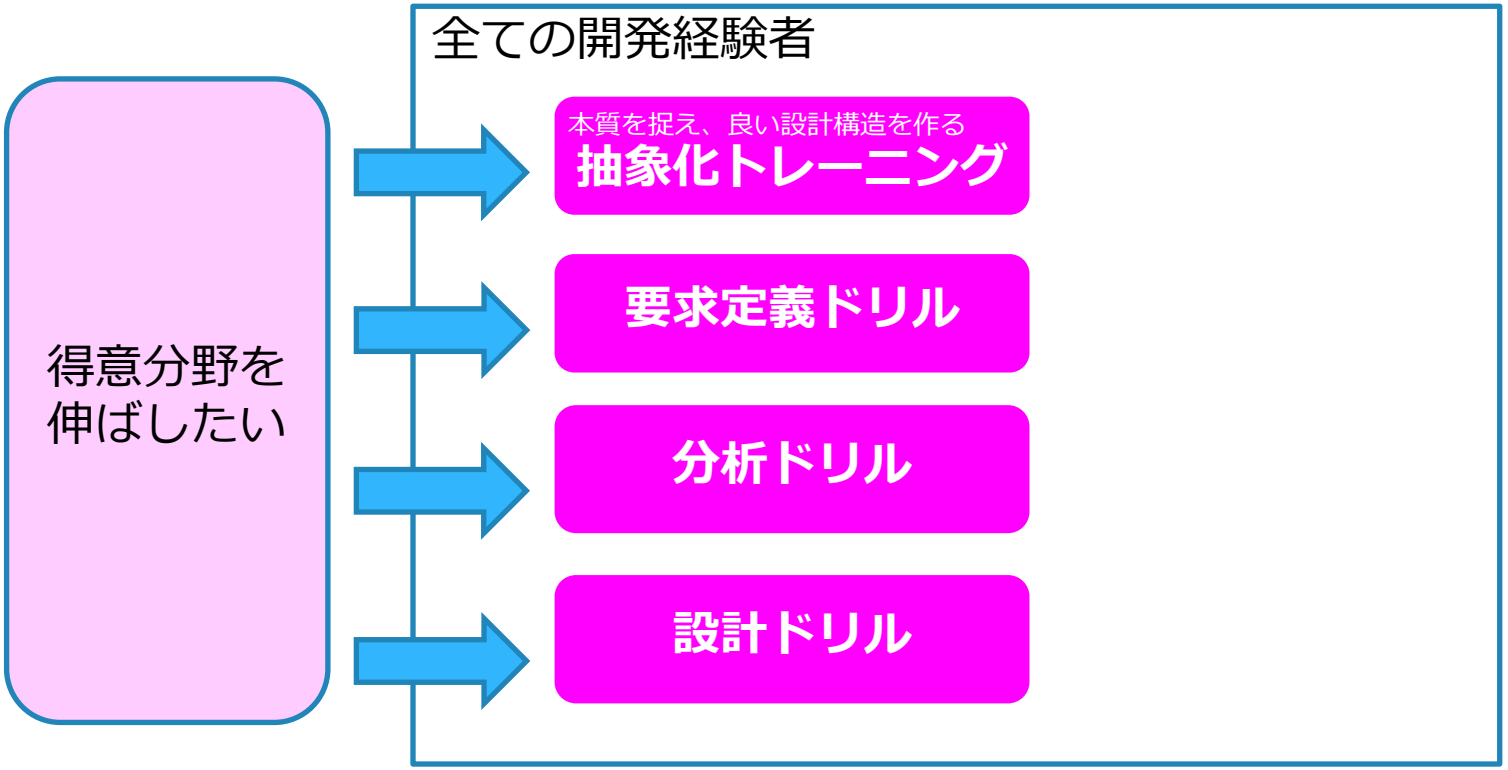


H. 実力診断セミナー

H. 実力診断

H. 実力診断

- 経験を増やすことで、スキルを向上させます
  - ドリル系は、５問の設問に答えていく方式です
    - 自分の実力もわかります
  - 抽象化トレーニングは、複数の題材をもとに、視点を定めて抽象化します



主な内容

要求定義ドリル

- 技術レポートを書く
- 再発防止策を書く
- 画面から仕様を定義する
- 要求記述から仕様を定義する
- インタビューから仕様を定義する
- 仕様を体系的に整理する

抽象化トレーニング

- 抽象化とは
- 制御仕様の抽象化
- 機能の抽象化
- 状態の抽象化
- イベントの抽象化
- 抽象化スキルを鍛える

設計ドリル

- フラットから構造化へ
- ヘッダファイルの役割
- 良い名称と悪い名称
- BOSS関数を作る
- シンメトリ
- 単方向依存

分析ドリル

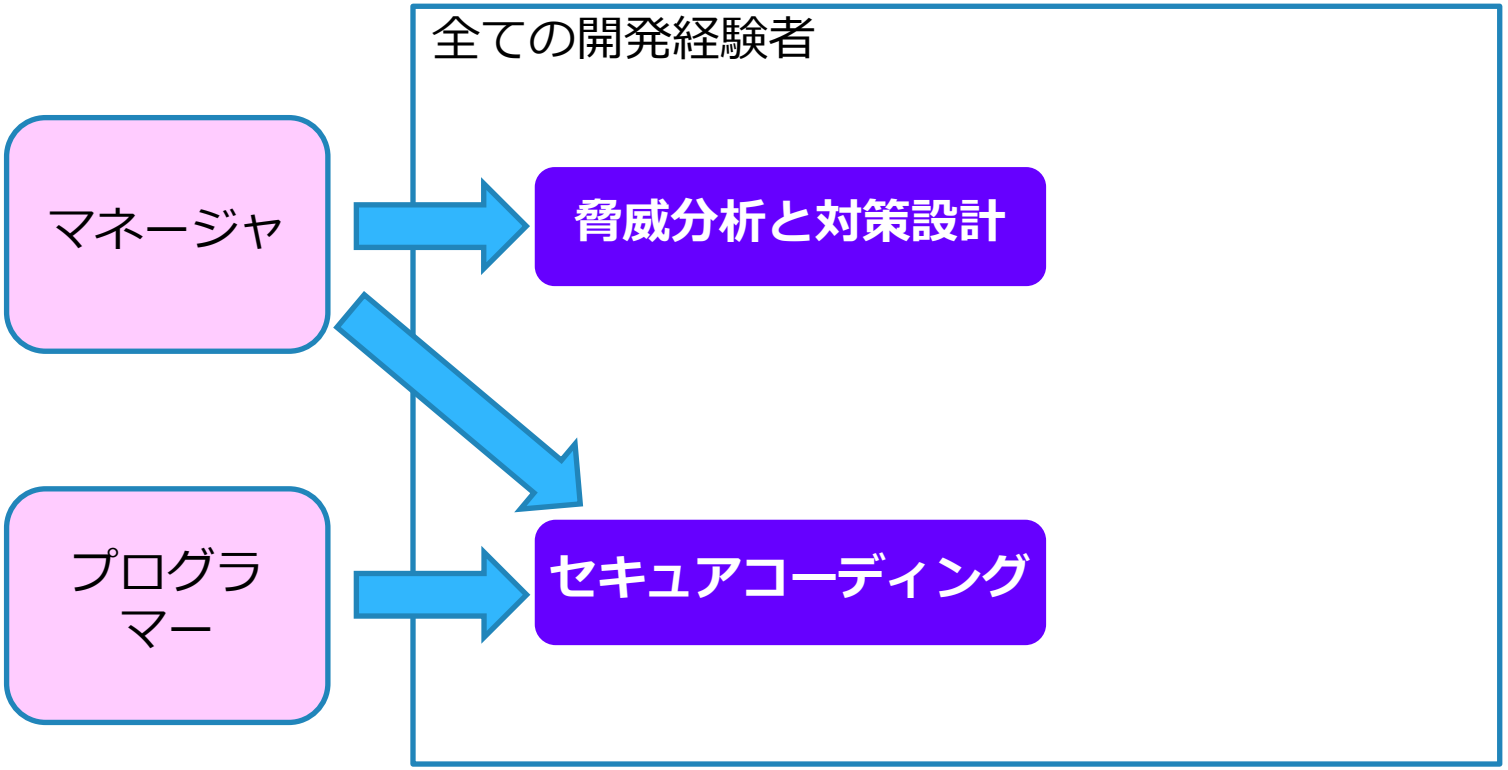
- 本質を捉える
- 捨象する
- 主要機能を見つける
- 重要なデータを定義する
- 状態を見つける
- 複数視点を統合する

I. 製品セキュリティ

I. 製品セキュリティ

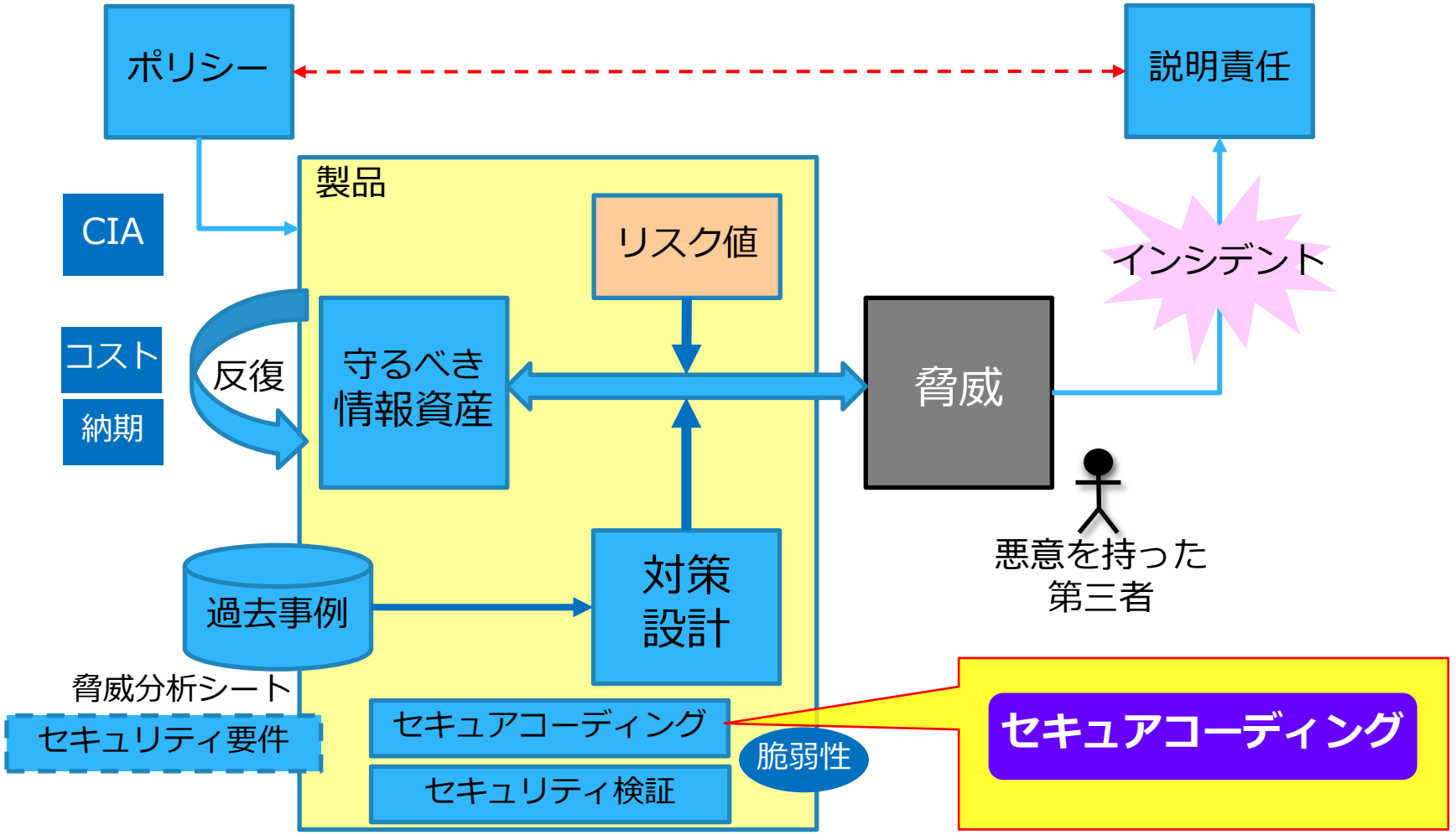
I. 製品セキュリティ

- セキュリティポリシーに従って脅威分析と対策設計の一連の流れを体験します
- セキュアコーディングでは、脆弱性のないソースコードの書き方を習得します



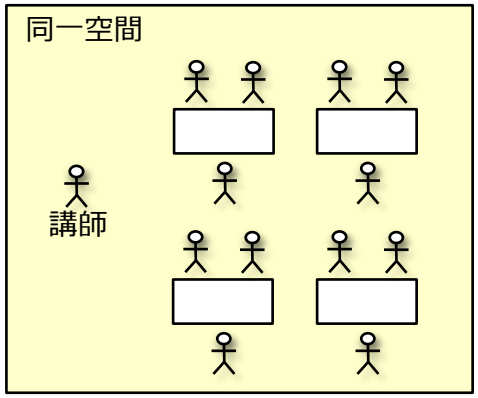
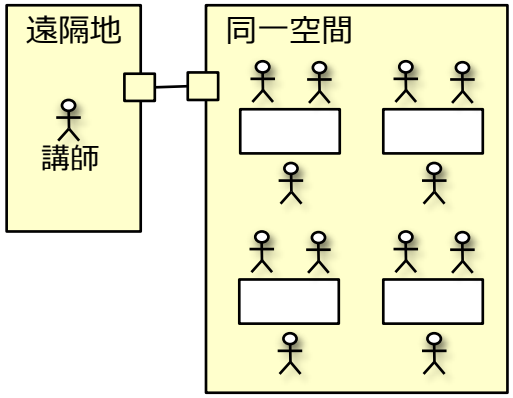
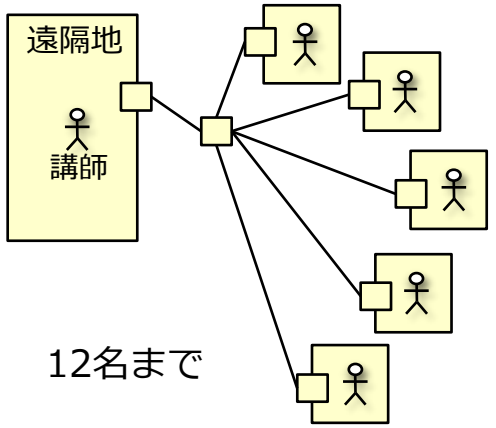
製品セキュリティの全体像

- **脅威分析と対策設計** セミナーでは、以下の用語と関連を理解します



４．開催形態（集合／オンライン）

3つの実施形態に対応

	集合セミナー	リモートセミナー	多地点セミナー
形態			
特徴	受講者と講師が対面	受講者は一つの部屋に集合し、講師のみ遠隔地	受講者と講師とも、個別に遠隔地（自宅含む）
リモートツール	（なし）	Teams （Zoom、WebEXでも可）	Teams （Zoom、WebEXでも可）
グループ演習	3人でひとチーム 模造紙／A3を作成	3人でひとチーム 模造紙／A3を作成	3人でひとチーム 討議しながら代表者がグループ成果物を作成
教材	当日、印刷物を配布	当日、印刷物を配布	前日までにpdfファイルで配布（印刷物は配布しない）
機材	当日、PCと実機を利用	当日、PCと実機を利用	前日までに各自のPCにシミュレータをインストール

実施形態別の進め方

	集合セミナー	リモートセミナー	多地点セミナー
1	講義	リモート講義	←（同左）
2	個人演習	←（同左）	←（同左）
3	グループ演習 ・模造図／A3用紙を作成	←（同左）	グループ討議 ・グループ成果物は代表者が作成し共有
4	講師コメント	典型的な成果物を紹介 ・各自で個人成果物と比較 ・全員でグループ成果物と比較	←（同左）
5	実機利用 ・LED点滅装置 ・LEGOロボット	シミュレータ利用 ・各自のPCへシミュレータをインストール	←（同左）
6	分担開発 ・インタフェース整合で分担（スケルトン駆動）	←（同左） ・ファイル共有の仕組みが必要	←（同左） ・インタフェース整合の方法を明記
7	質疑 理解度確認	リモート質疑	←（同左） 個人理解促進として、必要に応じて書籍を配布 ・構造化プログラミング、など

演習種別と実施形態

演習種別	特徴	実施形態	セミナー例
解答型	答えがある演習	集合セミナー オンラインセミナー 多地点セミナー	伝わる！コーディング 伝わる！仕様記述 いまさら聞けない 設計思考基礎 リバースモデリング
思考型	絶対解がない演習	集合セミナー オンラインセミナー	構造化モデリング[静的][動的] アーキテクチャ設計基礎／実践 リアーキテクティング
	実機演習は シミュレータで		設計技法基礎、設計図活用 分析設計テスト
実習型	リーダーシップ系	集合セミナー	アーキテクト育成[ものづくり] アーキテクト育成[ことづくり] アーキテクト育成[システムズ]
添削型	問に対する回答とその添削	集合セミナー オンラインセミナー 多地点セミナー	設計ドリル 分析ドリル 要求定義ドリル