IPA Software Engineering Center
Information-technology Promotion Agency, Japan

共通フレーム2013の概説

独立行政法人情報処理推進機構（IPA）
技術本部 ソフトウェア・エンジニアリング・センター（SEC）

研究員　室谷　隆

はじめに

ソフトウェアエンジニアリングとは

１．ソフトウェアの開発、運用、および保守における、システマティックであり、ディシプリン(*)に基づいた、定量的なアプローチの適用である。換言すれば、ソフトウェアへの工学の適用である。

２．１．で示したアプローチに関する研究である。とされている。

(*)ディシプリン：方法論に基いた教育・訓練によって形成された規律

つまり

体系化し、
それに従った手順を作成し作業し、
データを収集して、フィードバックすること。

松本吉弘訳　ソフトウェアエンジニアリング基礎知識体系－SWEBOK　2004－:オーム社　より

目 次

第1部　共通フレーム2013の概要

SEC
Software Engineering for Mo·No·Zu·Ku·Ri

１．共通フレームとは

２．共通フレーム2013の経緯

３．なぜ、プロセスが重要なのか

４．共通フレームの特徴

５．共通フレームの構造

６．共通フレームとガイダンスの見方

７．共通フレームのプロセス体系

８．共通フレーム2007との差異

９．修整（テーラリング）の適用について

１０．テーラリング方法

1．共通フレームとは（1/2）

SEC
Software Engineering for Mo·No·Zu·Ku·Ri

■共通フレームとは

ソフトウェアの構想から開発、運用、保守、廃棄に至るまでのライフサイクルを通じて必要な作業項目、役割等を包括的に規定した共通の枠組み。

何を実施するべきかが記述されている、

「ＩＴシステム開発の作業規定」である。

■その目的は

日本において、ソフトウェア開発に関係する人々（利害関係者）が、「同じ言葉で話す」ことが出来るようにするため。

1．共通フレームとは（2/2）

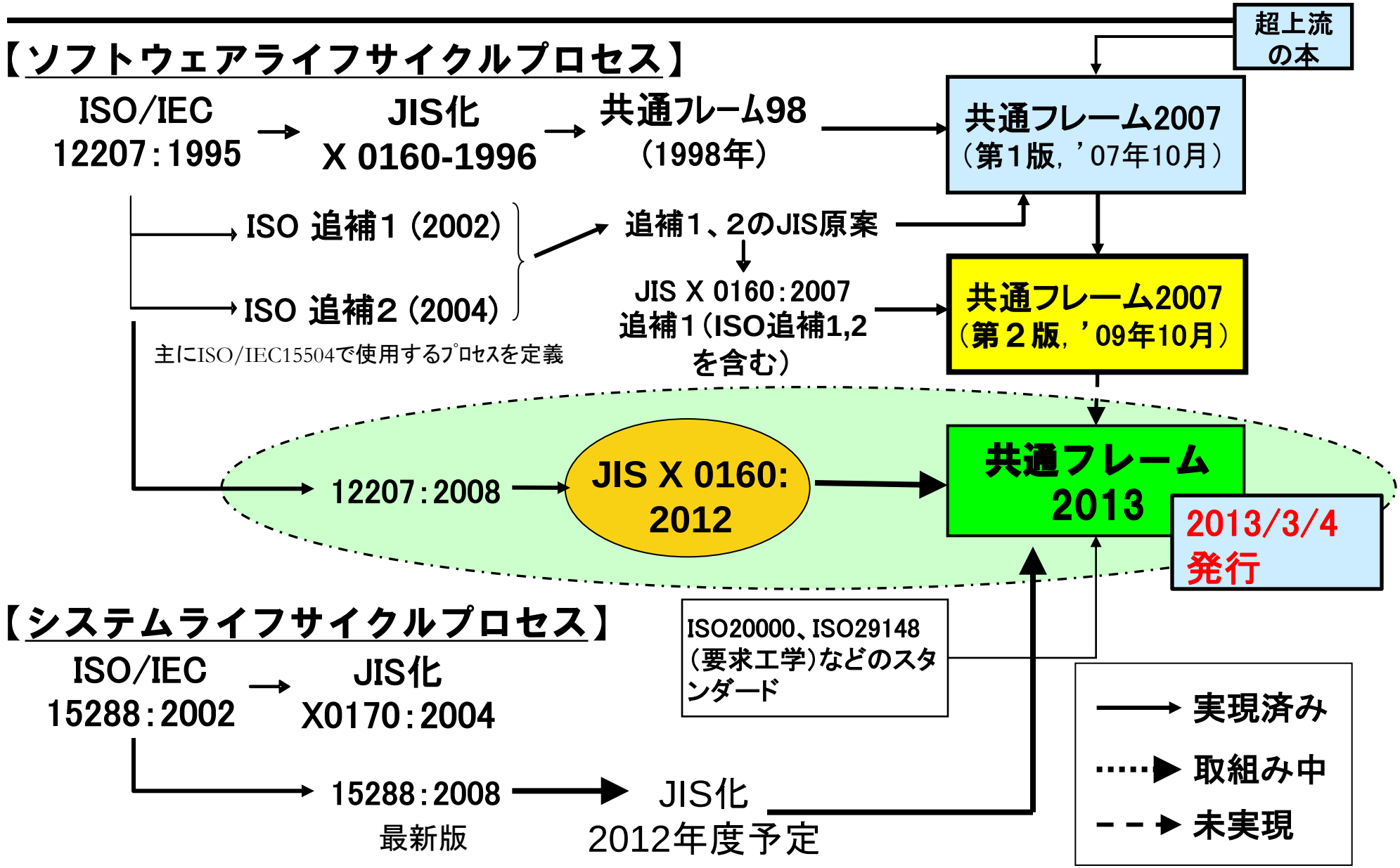
■作成者は

ユーザ企業、ベンダ企業、IPA/SEC、大学、経済産業省からなる開発プロセス共有化部会（2007年10月当時）である。

■ソフトウェア開発方法論との関係は

ウォーターフォール、スパイラル、プロトタイプ、アジャイル系
すべての開発方法論に共通したもの。

2. 共通フレーム2013の経緯



3. なぜ、プロセスが重要なのか

SEC
Software Engineering for Mo·No·Zu·Ku·Ri

■プロダクトの品質はプロセスの品質から
（工学（エンジニアリング）の基本）

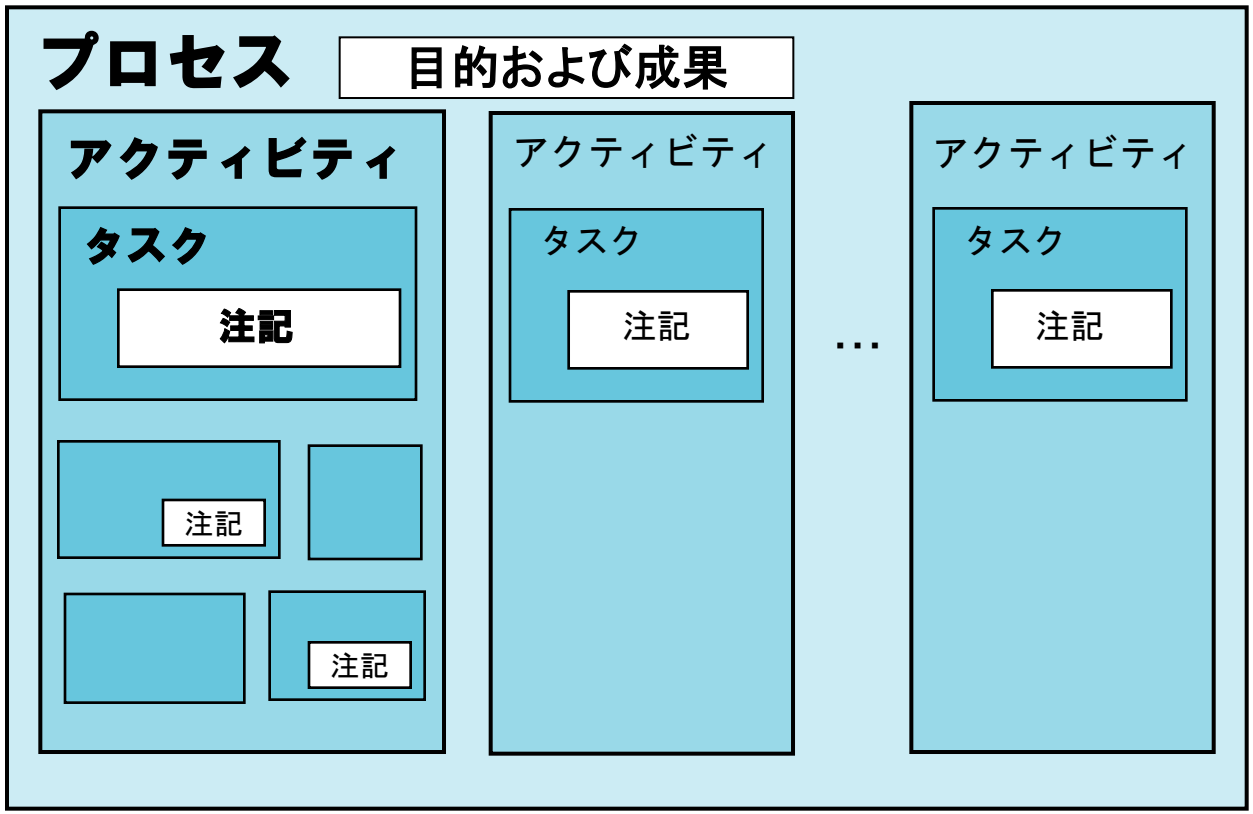
■プロセス：インプットをアウトプットに変換する，相互に関連する又は相互に作用する一連の活動（JIS Q 9000:2006）（処理する、加工する、手を加える）
活動を役割の観点でまとめている。
例　開発プロセス、運用プロセス、保守プロセス
What to do（何をするか）であり、
How to do（どのようにするか）は決めていない。

４．共通フレームの特徴

（１）超上流の重視
（２）モジュール性の採用
（３）責任の明確化
（４）責任範囲の明確化
（５）工程、時間からの独立性
（６）開発モデル、技法、ツールからの独立性
（７）ソフトウェアを中心としたシステム関連作業までを包含
（８）システムライフサイクルプロセスとの整合性
（９）文書の種類、書式を規定しない
（１０）修整（テーラリング）の採用

5. 共通フレームの構造

●次の図のように、4つの要素が階層化されている。



■ プロセス とは、

システム開発作業を役割の観点でまとめたもの。

■ アクティビティ とは、

相関の強いタスクをまとめたタスクの集合のこと

■ タスク とは、

アクティビティを構成する個々の作業のこと

■ 注記 とは、

タスクを構成する要素のこと。

例示としている。

6. 共通フレームとガイダンスの見方

（本体の形式）

2.3.5.2.1 システム適格性確認テストの準備

システムの適格性確認の各要件に対して，システム適格性確認テストを行うため，一連のテスト，テストケース（入力，出力及びテスト準備など）及びテスト手順を作成し，文書化する。開発者は，結合したシステムに対してシステム適格性確認テストの準備が終わっていることを確実にする。

テスト実施にあたって各種マスタファイルのデータ，トランザクションデータを作成し，テスト環境に登録する。

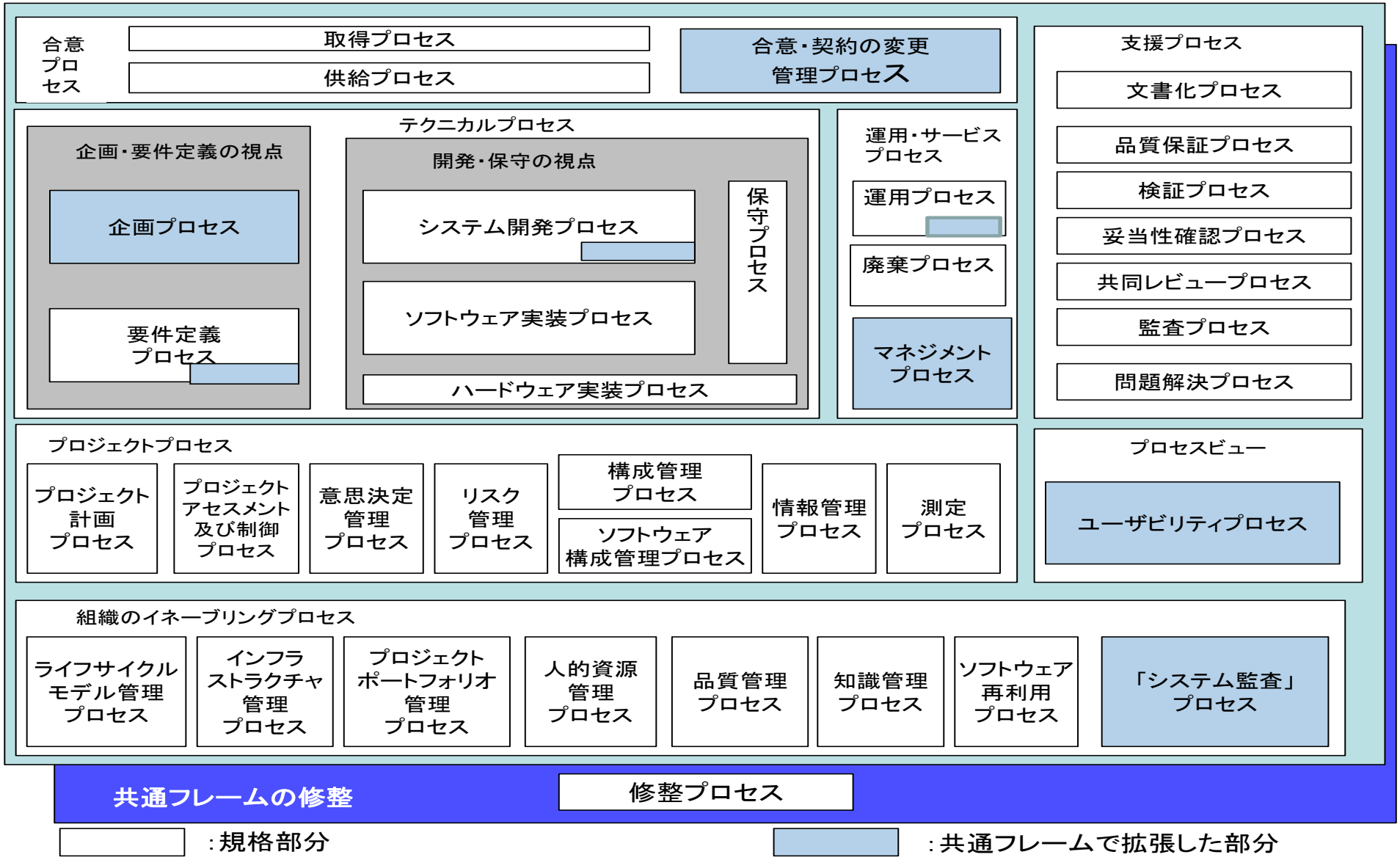
注記　変更したときにシステムを再テストするために適用する回帰テスト戦略を作成することが望ましい。

ガイダンス

2.3.5.2.1：データは、できる限り本稼働で用いるデータに近いものを設定する。現行システムのデータが存在する場合は，セキュリティを考慮した上で、それを移行して利用する。

（青色の囲み）　共通フレーム定義体 を表す。
（文字種）　国際標準：**太字**、日本で追加、変更した部分：細字。
（ガイダンス）　日本で追加した解説を表す。

7. 共通フレーム2013のプロセス体系



8. 共通フレーム2007との差異(1/18)

■国際規格ISO/IEC 12207（JIS X 0160）の改訂

（１）プロセスの追加と変更

JIS X0160:1996（追補1含む）からJIS X0160:2012への改訂に伴い，以下の8つのプロセスが追加された。

- プロジェクトポートフォリオ管理プロセス
- 品質管理プロセス
- 意思決定プロセス
- リスク管理プロセス
- 構成管理プロセス
- 情報管理プロセス
- 利害関係者要求定義プロセス（要件定義プロセス）
- 実装プロセス

8. 共通フレーム2007との差異 (2/18)

（ａ）プロジェクトポートフォリオ管理プロセス

組織の優先順位づけや成果を明確化し，組織の戦略的目標を満たすための資源を割当てるとともに，プロジェクトが計画に則って進行しているかどうかを評価する。

（ｂ）品質管理プロセス

製品及びサービス，ならびにそれらの作成プロセスが組織の品質目標に合致し，顧客満足を達成することを保証する。

（ｃ）意思決定プロセス

代替手段がある場合，プロジェクトとして最も有益な進路を選定する。

（ｄ）リスク管理プロセス

リスクを継続的に識別し，分析し，取り扱い，監視する。

8. 共通フレーム2007との差異 (3/18)

（ｅ）構成管理プロセス

システムの構成管理。

（ｆ）情報管理プロセス

システムライフサイクルの期間中，関連する情報（安全な，妥当な，適時の，機密の）を提供する。

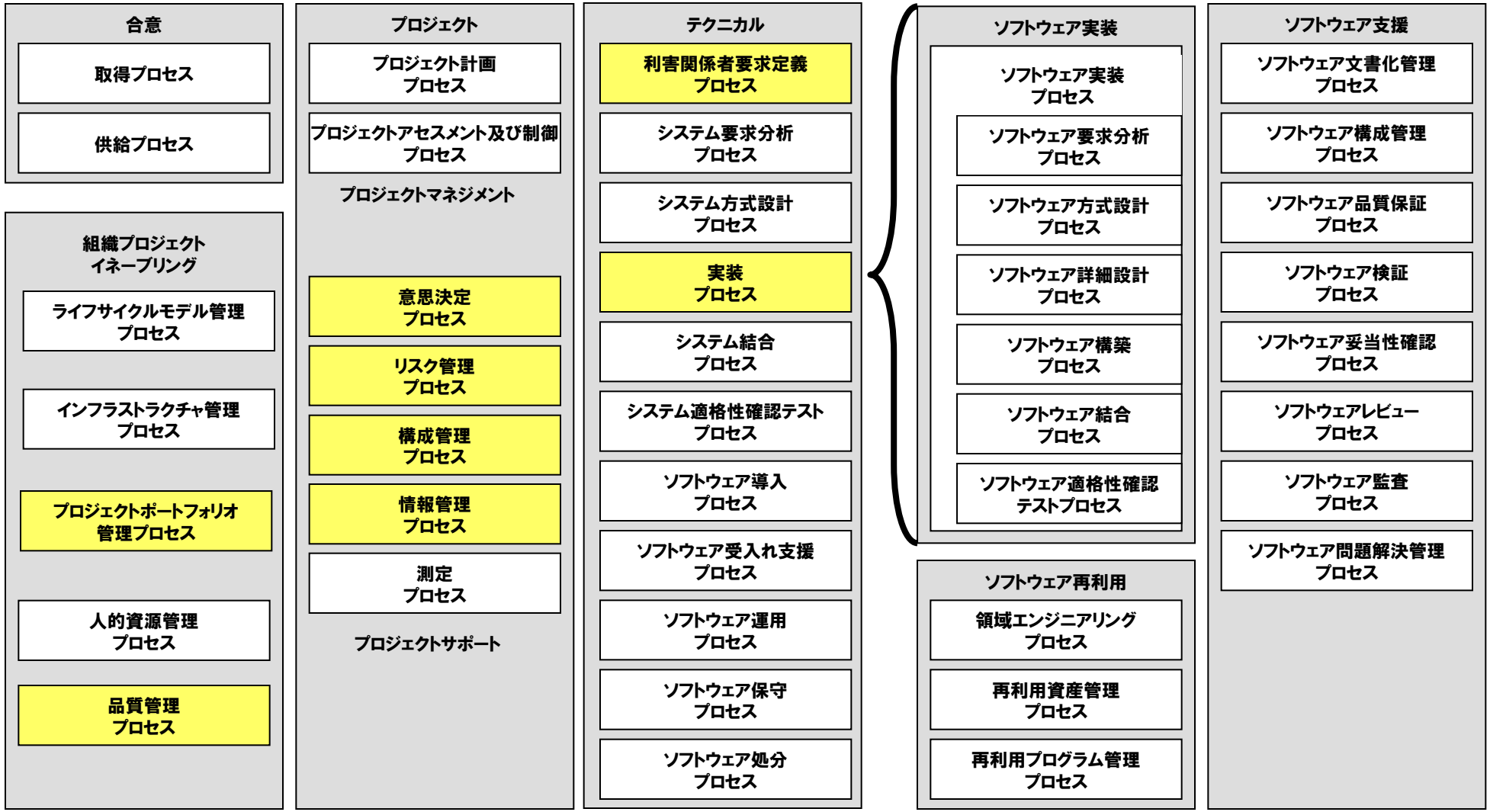
（ｇ）利害関係者要求定義プロセス（要件定義プロセス）

組織として、システムが実現する仕組みに係る要件を定義する。

（ｈ）実装プロセス

組織として、ソフトウェア開発を行う。

8. 共通フレーム2007との差異 (4/18)
ISO/IEC12207:2008(JIS X 0160:2012)の構成



1995版と同等　　新規追加

8. 共通フレーム2007との差異 (5/18)

（２）プロセスの名称変更、アクティビティからプロセスへの変更

- 改善プロセスをライフサイクルモデル管理プロセスに名称変更
- 環境整備プロセスをインフラストラクチャ管理プロセスに名称変更
- 資産管理プロセスを再利用資産管理プロセスに名称変更
- 測定アクティビティを測定プロセスに変更
- システム又はソフトウェア廃棄アクティビティを廃棄プロセスに変更
- 開発プロセスのアクティビティをプロセスに変更
- ソフトウェアコード作成及びテストアクティビティをソフトウェア構築プロセスに名称変更

8. 共通フレーム2007との差異 (6/18)

SEC Software Engineering for Mo·No·Zu·Ku·Ri

■共通フレーム2007から変更、拡張した点

大きくは次の３点となる

（１）ベースとなるJIS規格の変更

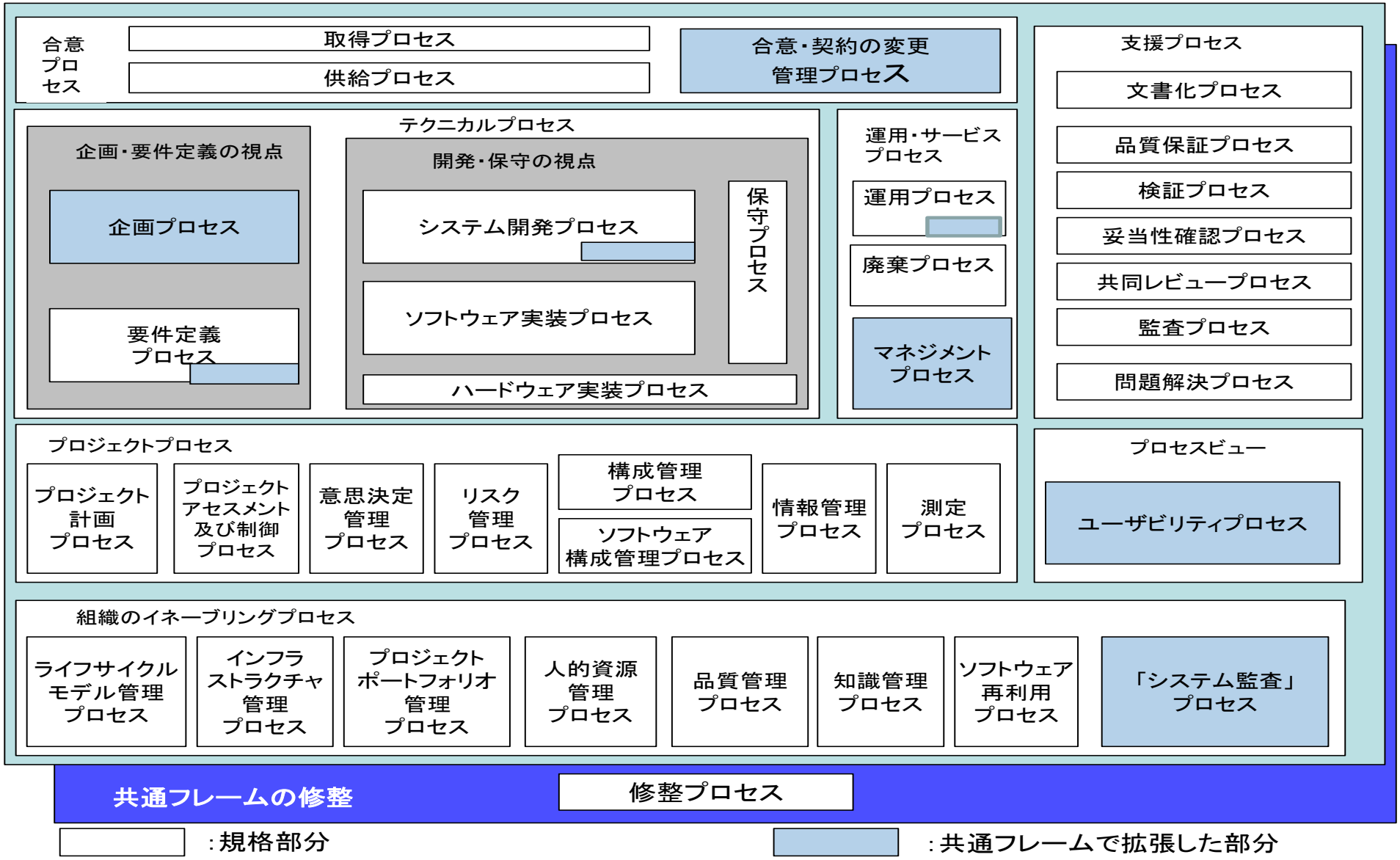
JIS X 0160:1995+2007からJIS X 0160:2012へ

（２）システム開発のプロセスを明示的に取り入れ

共通フレーム2007では，ソフトウェア開発中の一部分にシステムの概念が入っていた程度であったものを，システム開発プロセスという括りで明確にした。

（３）運用プロセスの強化とサービスマネジメントプロセスの追加

共通フレーム2013のプロセス体系



8. 共通フレーム2007との差異 (7/18)

SEC Software Engineering for Mo·No·Zu·Ku·Ri

■共通フレームで変更、拡張した点

（１）JIS X 0160に追加、変更、拡張

- 企画プロセス
- 要件定義プロセス
- 合意・契約の変更管理プロセス
- サービスマネジメントプロセス（と運用プロセス）
- ユーザビリティプロセスビュー
- ハードウェア実装プロセス
- システム受入れ支援プロセス
- システム導入プロセス
- 知識管理プロセス
- 「システム監査」プロセス

8. 共通フレーム2007との差異 (8/18)

SEC Software Engineering for Mo·No·Zu·Ku·Ri

- 他のプロセスのアクティビティ，タスクに関しても実際の開発及び取引に必要な作業項目の定義を追加している。

（ａ）企画プロセス

企画プロセスの目的は，経営・事業の目的，目標を達成するために必要なシステムに関係する要件の集合とシステム化の方針，及び，システムを実現するための実施計画を得ることである。システム化構想の立案とシステム化計画の立案プロセスが含まれる。

システム化構想の立案プロセスでは，経営課題を解決するための新たな業務とシステムの構想を立案する。

システム化計画の立案プロセスでは，システム化構想を具現化するための，システム化計画及びプロジェクト計画を具体化し，利害関係者の合意を得る。

8. 共通フレーム2007との差異 (9/18)

（ｂ）要件定義プロセス

ISO/IEC 12207:2008（JIS X0160:2012）では利害関係者要求定義プロセスが新設され，取得者の業務要件，ならびに取得者がシステムに求める要件（機能要件，非機能要件）を明確にするプロセス。

共通フレーム2007で新設した要件定義プロセスがこれに当たる。

共通フレーム2013では利害関係者要求定義プロセスの内容を要件定義プロセスとして採用し，運用シナリオの概念をISO/IEC/IEEE 29148（要求工学）から取り入れ拡張。

8. 共通フレーム2007との差異（10/18）

（ｃ）合意・契約の変更管理プロセス

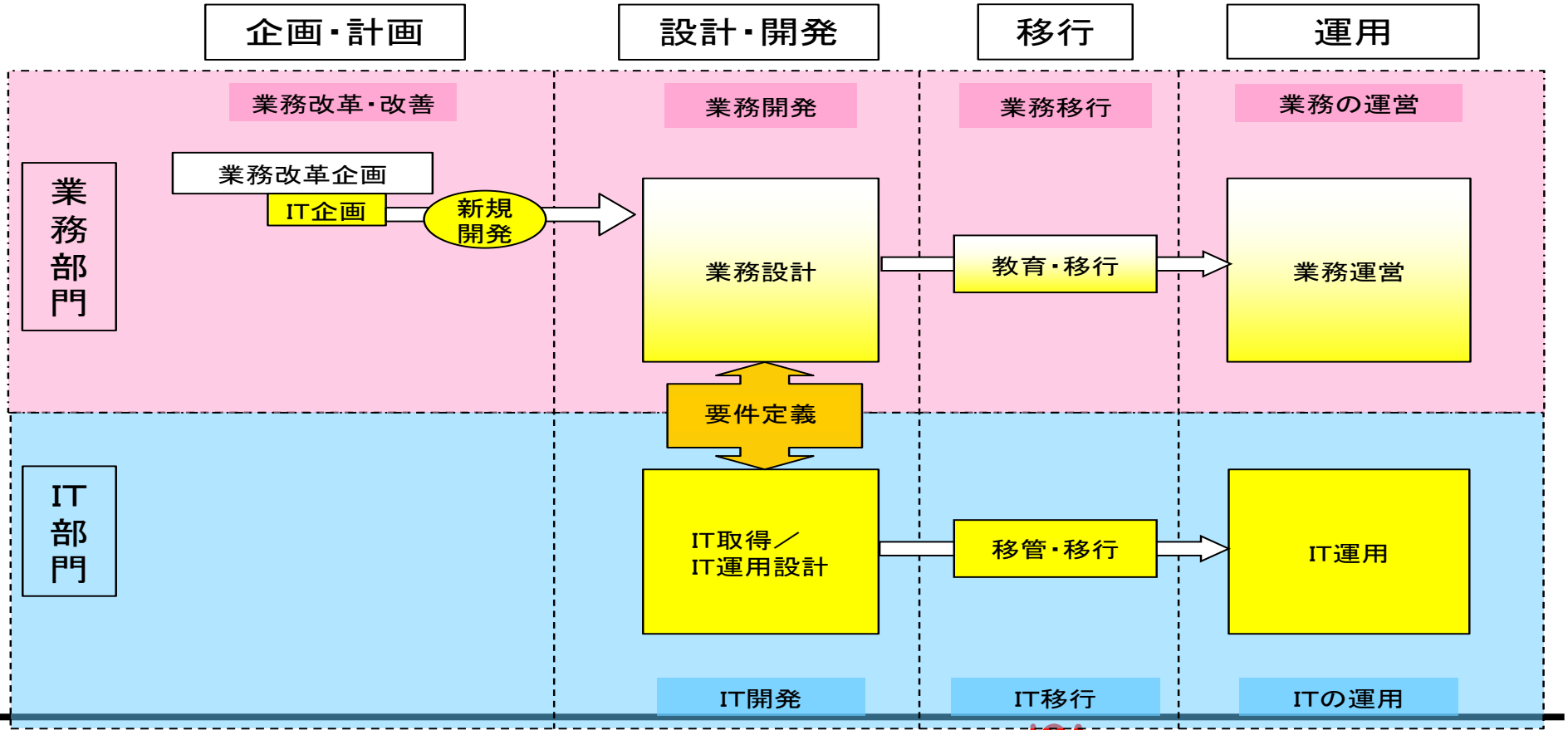
共通フレーム2007で追加したプロセスであり、ISO/IEC 12207:2008（JIS X0160:2012）に提案し，附属書F（参考資料）として採用されたものである。

共通フレーム2013では，この参考資料とされている附属書FをJIS翻訳のまま採用するのではなく，国際規格に提案した原案（共通フレーム2007で追加したプロセスの日本語文）を合意・契約の変更管理プロセスとした。

8. 共通フレーム2007との差異（11/18）

（ｄ）サービスマネジメントプロセスと運用プロセス

共通フレーム2007では，主に業務システムを取得するまでのプロセスを中心に，取得者と供給者の作業を記述してきた。運用は取得後の後工程の位置付けであった。



8. 共通フレーム2007との差異 (12/18)

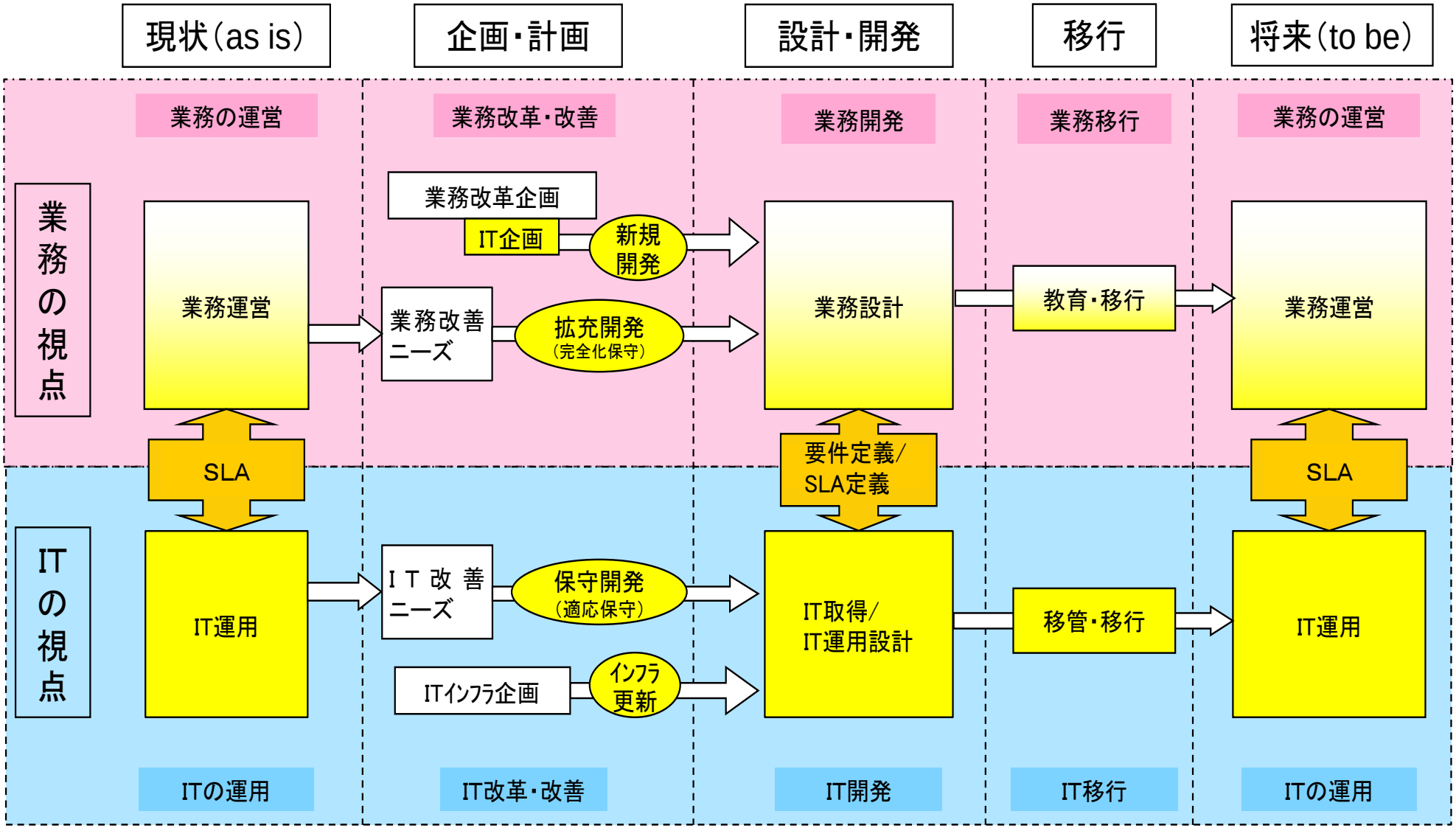
業務システムは，取得しただけでは何の価値も生まない。システムを運用し，業務で利用されて初めて価値を生む。

経営者は，システム取得を一過性の投資としてIT部門に任せるのではなく，業務運用あるいは改善の一環としてとらえ，事業の発展に合わせてシステムを育てるという見方をすることが重要になってくる。

運用・サービスプロセスを充実させ，運用を重視した開発が可能となるようタスクやガイドの一部を更新。

特にサービス運用については，国際規格ISO/IEC 20000（JIS Q20000）が広く受け入れられてきていることから，ISO/IEC 20000（JIS Q20000）を既に導入している企業が共通フレームとの整合を図れるようにISO/IEC 20000（JIS Q20000）のプロセスとのインタフェースとなるサービスマネジメントプロセスを新設した。

現在の運用の位置付け



8. 共通フレーム2007との差異(13/18)

SEC
Software Engineering for Mo·No·Zu·Ku·Ri

（ｅ）ユーザビリティプロセスビュー

ユーザビリティプロセスはISO/IEC 12207 Amd1:2002（JIS X0160:2007　追補1）で採用されたプロセスである。

ISO/IEC 12207:2008（JIS X0160:2012）においてはプロセスビュー（注）の一例として定義されるにとどまっている。

共通フレーム2013では，共通フレーム2007との継続性及びユーザビリティの重要性を考慮して，共通フレーム2013本体に組み入れた。

（注）プロセスビュー：特定の関心事に焦点を当て，その履行や達成に必要となる目的や成果，“既定の”プロセス，アクティビティ，タスクを示すための表現法である

8. 共通フレーム2007との差異 (14/18)

SEC Software Engineering for Mo·No·Zu·Ku·Ri

（ｆ）ハードウェア実装プロセス

ITで言うシステムは，ソフトウェアとハードウェアが組み合わさったもの。

共通フレーム2007までは，ソフトウェア開発プロセスの中にシステムの概念を入れ，両者が混在している状態。

共通フレーム2013では，この混在していたプロセスをシステム開発，ソフトウェア実装として明確に分離し、ソフトウェアと対を成すハードウェア関連のプロセスを定義。

ハードウェア実装プロセスは大枠の定義体のみであり，詳細のプロセスは定義していない。

8. 共通フレーム2007との差異（15/18）

（ｇ）システム受入れプロセス

（ｈ）システム導入プロセス

システム開発とソフトウェア実装を明確に分離した結果，ISO/IEC 12207:2008（JIS X0160:2012）で規定されているプロセスでは，システム開発から見て不足しているプロセスがあることが判明。

システム受入れプロセスとシステム導入プロセスである。この不足しているプロセスを新たに作成。

（ｉ）知識資産管理

規格上は人的資源管理プロセスのアクティビティとして定義されている。

共通フレーム2013では，知識資産管理を重要なプロセスと位置づけ，人的資産管理プロセスから独立させて知識資産管理プロセスとした。

8. 共通フレーム2007との差異(16/18)

（ｊ）ソフトウェア支援プロセスを支援プロセスに変更

共通フレーム2013はシステム開発とソフトウェア実装の統合版。ISO/IEC 12207:2008（JIS X0160:2012）で定義されているソフトウェア支援プロセスはソフトウェア用であり，システム開発には使わないように見えてしまう。これを避けるため，ソフトウェア支援プロセスの名称から「ソフトウェア」を取り，支援プロセスとした。

（ｋ）「システム監査」プロセスの名称変更と内容の改訂

支援プロセスの監査プロセスがソフトウェアだけでなく、システムの監査も行うことになると２つの監査プロセスの位置付けが曖昧になる。このため日本のシステム監査基準、システム管理基準に基づいた監査を実施するプロセスとして明確にするため「」を付けた名称とした。また内容を大幅に見直し、システム開発、ソフトウェア実装（開発）のプロセスが変更になっても監査の対象が明確になるようにした。

8. 共通フレーム2007との差異（17/18）

（Ｉ）ソフトウェアレビュープロセスを共同レビュープロセスに変更

ソフトウェア支援プロセスを支援プロセスに変更したことに伴い，レビューはソフトウェア開発だけに適用するのではなく，システム開発にも適用することになった。ソフトウェアレビューの名称のままでは誤解を招く可能性があるため，共通フレーム2007で使用していた共同レビュープロセスに変更した。

8. 共通フレーム2007との差異(18/18)

SEC
Software Engineering for Mo·No·Zu·Ku·Ri

（m）ISO/IEC 29148（Requirements Engineering）の取り入れ

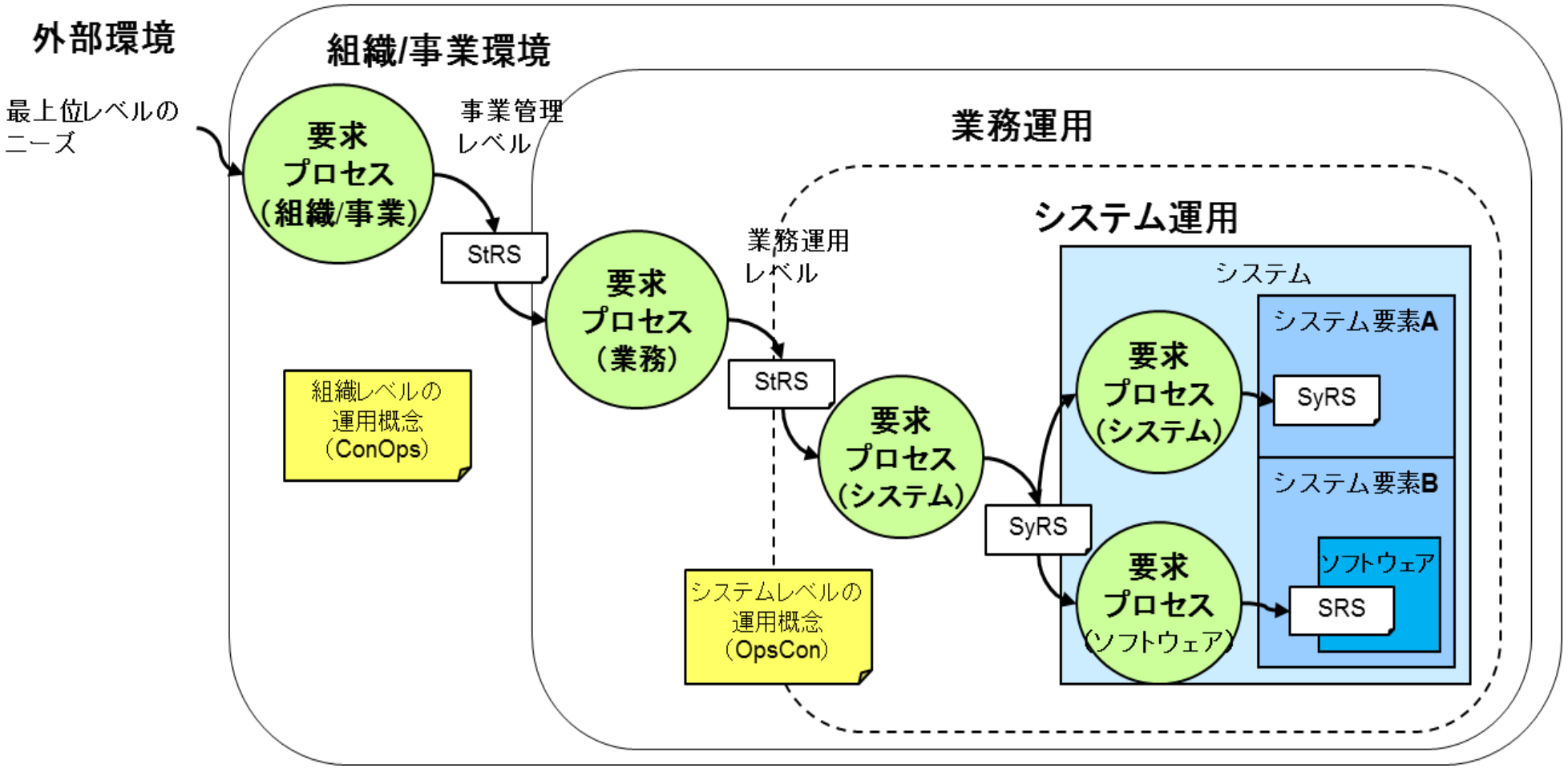
ISO/IEC 29148（Requirements Engineering）は2011年に発行され，2013年度にJIS化される予定。

規格の開発に当たり，日本から「超上流」や「それを取り込んだ共通フレーム2007」を提案。

要件は事業要件，業務要件，システム要件，ソフトウェア要件の4階層から成り、要件定義を行う際のプロセスを規定したものである。

この規格の中から「事業レベルの運用概念(Concept of Operations)」と，「システムレベルの運用概念(System Operational Concept)」という要件定義に必要なものを取入れ，要件定義プロセスや運用プロセスのアクティビティ，タスクとして追加。

「要件」の4階層 (ISO/IEC/IEEE 29148より)

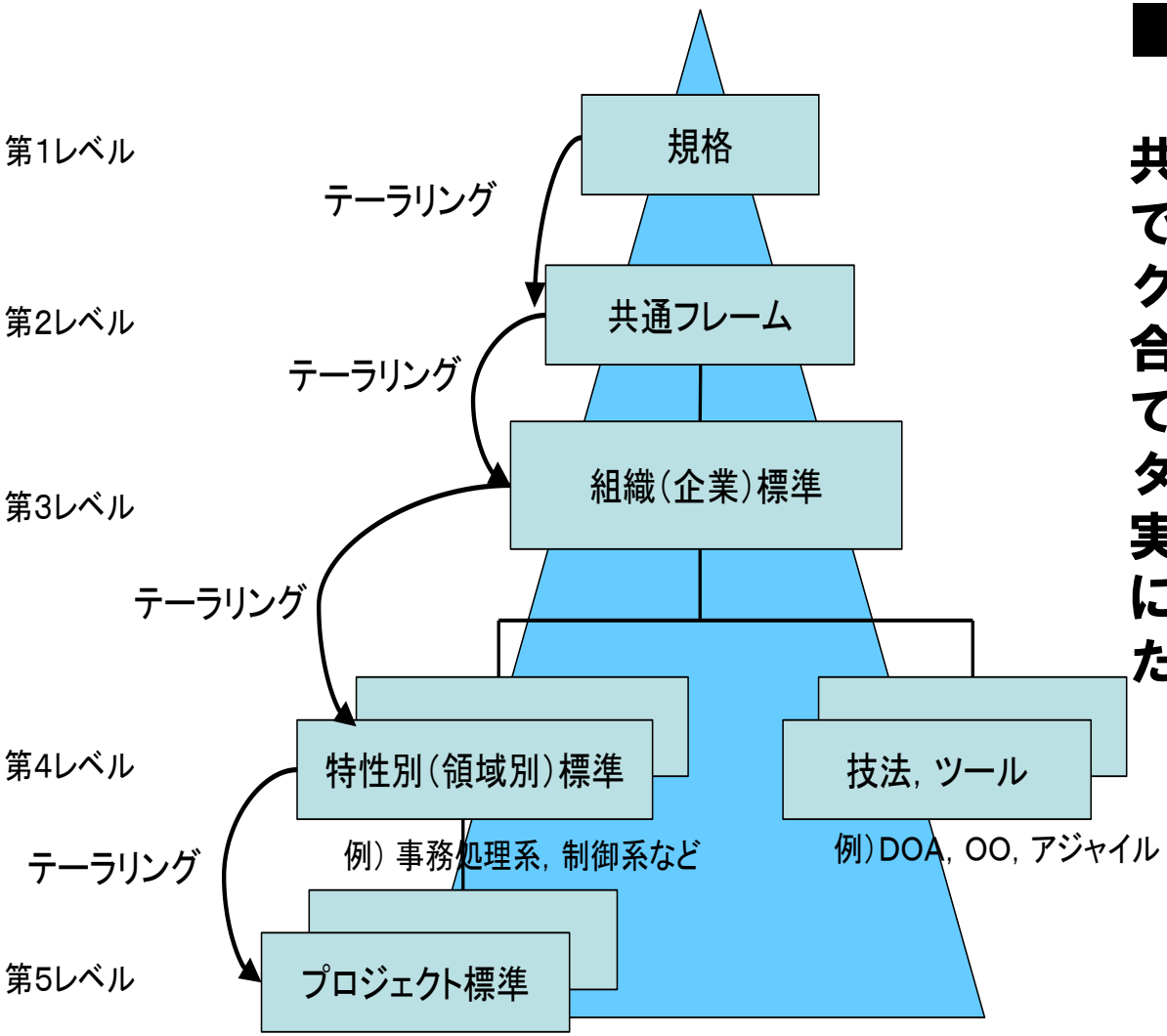


StRS：ステークホルダ要求　SRS：ソフトウェア要求

SyRS：システム要求

ISO/IEC29148　JIS原案より

9. テーラリング(修整)の適用について(1/2)



■修整(テーラリング)とは

共通フレームをそのまま適用するのではなく、組織(企業)やプロジェクトの特性(例えば開発モデル)に合わせて、共通フレームで規定されているプロセス/アクティビティ/タスクを取捨選択したり、繰り返し実行できるように、又は複数を一つに括って実行できるように組み替えたりする作業をいう。

(注1) DOA : データ中心のアプローチ
(注2) OO : オブジェクト指向の方法論, 技法など

9. テーラリング(修整)の適用について(2/2)

SEC Software Engineering for Mo·No·Zu·Ku·Ri

■ テーラリングのポイント

（１）「共通フレームで規定されている事を、すべて実施しなければならない」ということではない。

（２）「共通フレームで規定されている事」を、妥当と判断した場合には、省略してもよい。

（組織（企業）標準やプロジェクト標準に加えなくてもよい、ということ）

（３）「共通フレームで規定していないこと」を、組織（企業）標準やプロジェクト標準に追加してもよい。

→ 組織やプロジェクトの特性に合わせて、できるだけ最適と思われる作業の組み立て（「プロセス設計」）を行うために必要な活動が、テーラリングである。

10. テーラリング方法(1/5)

(1)作業工程を定義する

・時間軸(管理の区切り)を取り入れて、組織やプロジェクトの作業に必要なプロセス、アクティビティ、タスクを時間軸にマッピングして工程定義を行う。

・他のプロセス、アクティビティ、タスクとの関連を時間軸で表現する。(手順を決める)

ー特に複数の企業が開発に携わる場合、当該工程に含まれるアクティビティやタスクを詳細に定義する。このことにより、言葉の統一が図られ認識のズレを防ぐことができる。

ー開発規模や特性に応じて、工程の中のアクティビティやタスクをまとめたり、細分化したり、また削除したりする。

ー支援プロセス(共同レビュー、検証など)を手順の中に盛り込む。

ー運用プロセスの移行・準備作業は、開発工程が終了した後の運用工程でから始めるのではなく、開発工程内で実施する。

10. テーラリング方法(2/5)

（２）作業成果物を決める

工程のアウトプットとしての作業成果物を決める。

（３）開発モデルを選択する

・開発モデルに依存していないため、プロジェクトの特性に応じた開発モデルを選択し、共通フレームにあるタスクを組み立てる。

－プロジェクト全体では、ウォーターフォールモデルを採用するが、企画・要件定義段階では、繰り返し型や一部プロトタイピング型の開発モデルを使ってシステム化の実現性を調査する。

➢ 開発モデルが異なっていても、実施するタスクは同じである。どの時点でどう実施するのかの違いである。

10. テーラリング方法(4/5)

（４）プロセスの利用者を具体化する

・共通フレームは、各プロセスの実施をどういった立場や資質の人間がなすべきかを適用主体者として定義している。

・実際の利用では、これを参考に組織から利用者を選定する必要がある。

ー企画プロセスの利用者は企画者であるが、実際の組織に当てはめると、業務部門であったり、企画部門であったりする。

・誰の責任で実施すべきか、どのタスクを誰がいつ実施すべきかを、組織、プロジェクト、開発モデルの特性に合わせる。

➢ 各プロセスには、それぞれ「プロセス開始の準備」というアクティビティが定義されているので参照されたい。

10. テーラリング方法(4/5) 適用例

- 外部委託した場合
  - 同じ工程名でも、実施内容が異なる。
  - 同じ実施内容でも、工程名称が異なる。
- このような場合、共通フレームの用語を使い、お互いの認識を一致させる。
- また、複数ベンターを使う場合も、全てのベンダーに同じ用語を使ってもらう。

工程名称	要件定義	外部設計		内部設計	コーディング／テスト	結合テスト	システムテスト
共通Fのプロセス、アクティビティ、タスク	要件定義	SYS要件定義 SW要件定義	SYS方式設計 SW方式設計	SW詳細設計	コーディング／テスト	SW結合／SW適格性確認テスト	SYS結合／SYS適格性確認テスト／運用テスト
A社	要件定義		外部設計	内部設計	プログラミング	SWテスト	システムテスト
B社	要件定義	基本設計		詳細設計	製造	テスト	結合テスト

第１部　終わり

第2部　日本独自のプロセス拡張のねらい

１．プロセス拡張のねらい

２．企画プロセスと要件定義プロセス

３．超上流とは？

４．もし超上流を軽視したら？

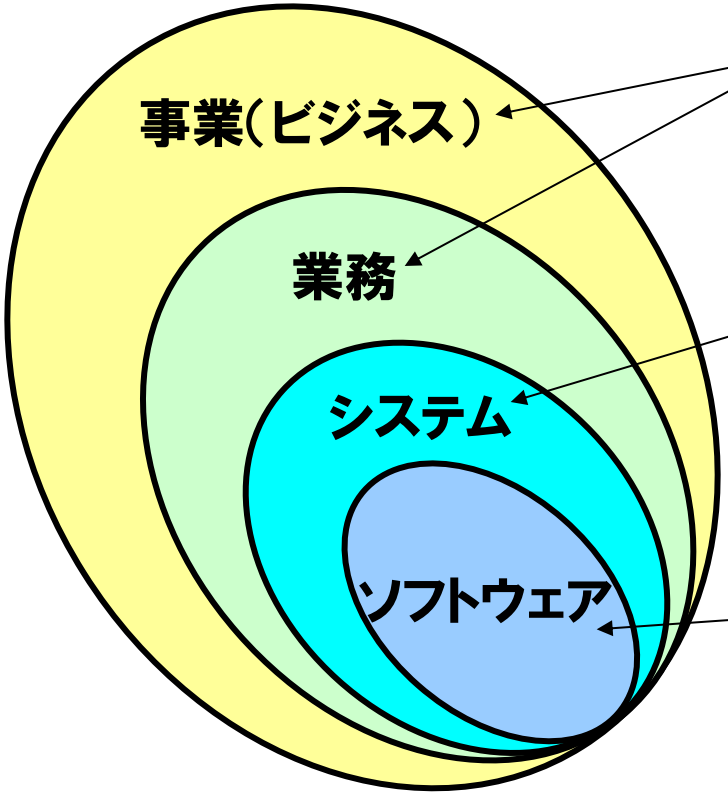
５．超上流でのトラブルの要因は？

６．共通フレームに含まれている主な考え方

1．プロセス拡張のねらい

ＩＴシステムは、事業（ビジネス）又は業務で使われるために開発される。

事業／業務における利用目的を明らかにし、その利用目的に応じて、システムに対する要求事項を定義することが非常に重要である。ここを疎かにしてしまうと、利用目的が曖昧となる。結果、「使い勝手の悪いシステム」や「利用されないシステム」等が出来上がってしまう恐れがある。共通フレームはこの考えを導入した。



事業又は業務レベル全体におけるシステム利用（人による活動も含む）に対する要求事項を明確に定義する。

システム（HW＋SW）に対する要求事項を定義する。

ソフトウェアに対する要求事項を定義する。

2. 企画プロセスと要件定義プロセス

■ 開発に入る前の「要求品質の確保」

システムは、事業（ビジネス）を実現するために開発される。
すなわち、開発に入る前の要求品質を確保することが重要になってくる。
このため、「超上流」と呼んでいる「企画」「要件定義」のプロセスが追加されたのである。

3. 超上流とは？

■ＳＥＣは、「共通フレーム2007」発行前に以下の書籍を刊行している。

- 「**経営者が参画する要求品質の確保**
  ～ 超上流から攻めるIT化の勘どころ ～」
  （第１版：2005年、第２版：2006年）
  ⇒ これ以降、本資料では『超上流の本』と呼ぶ。

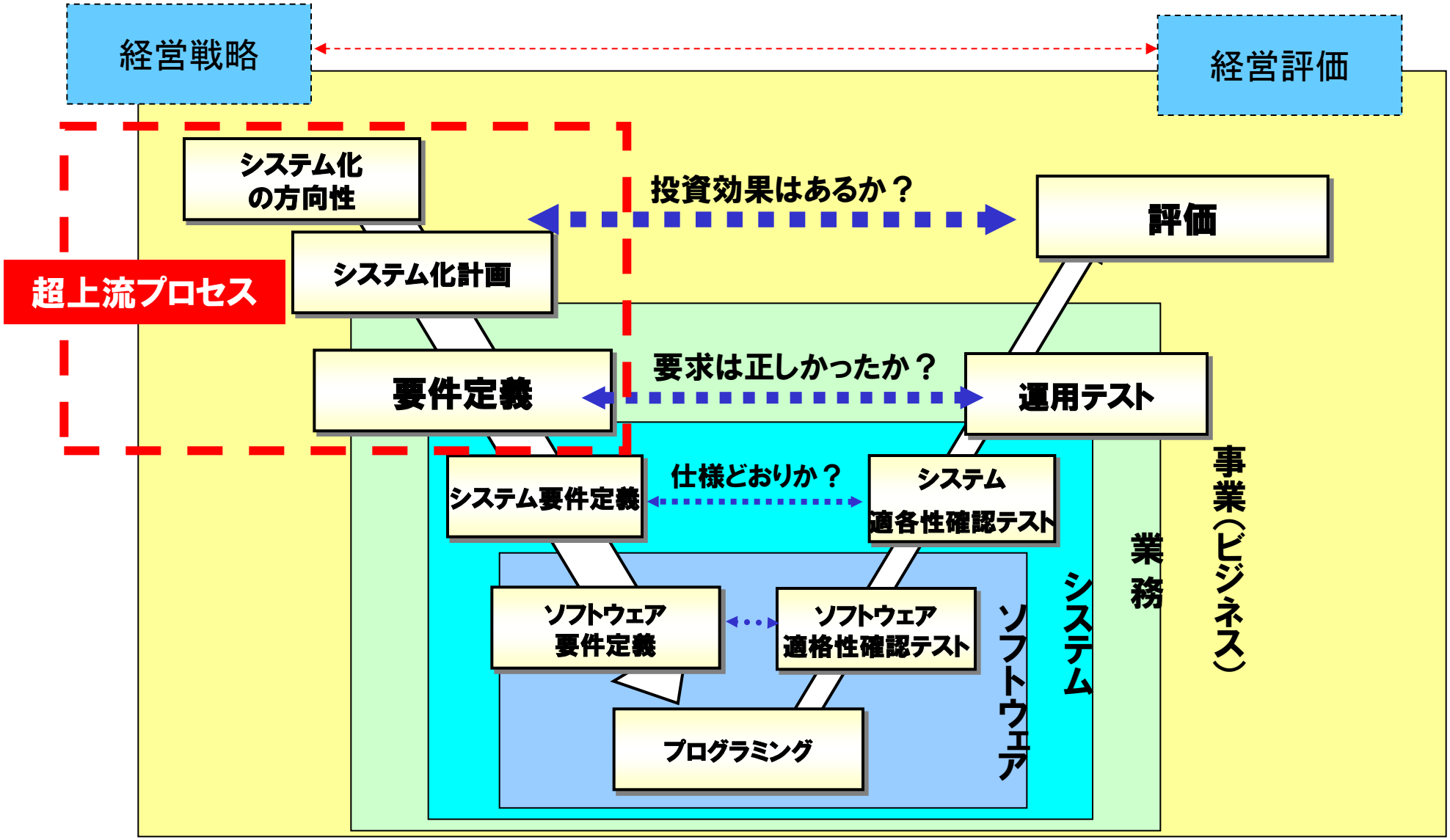
【この本のポイント】

① 超上流の重視を説いている。

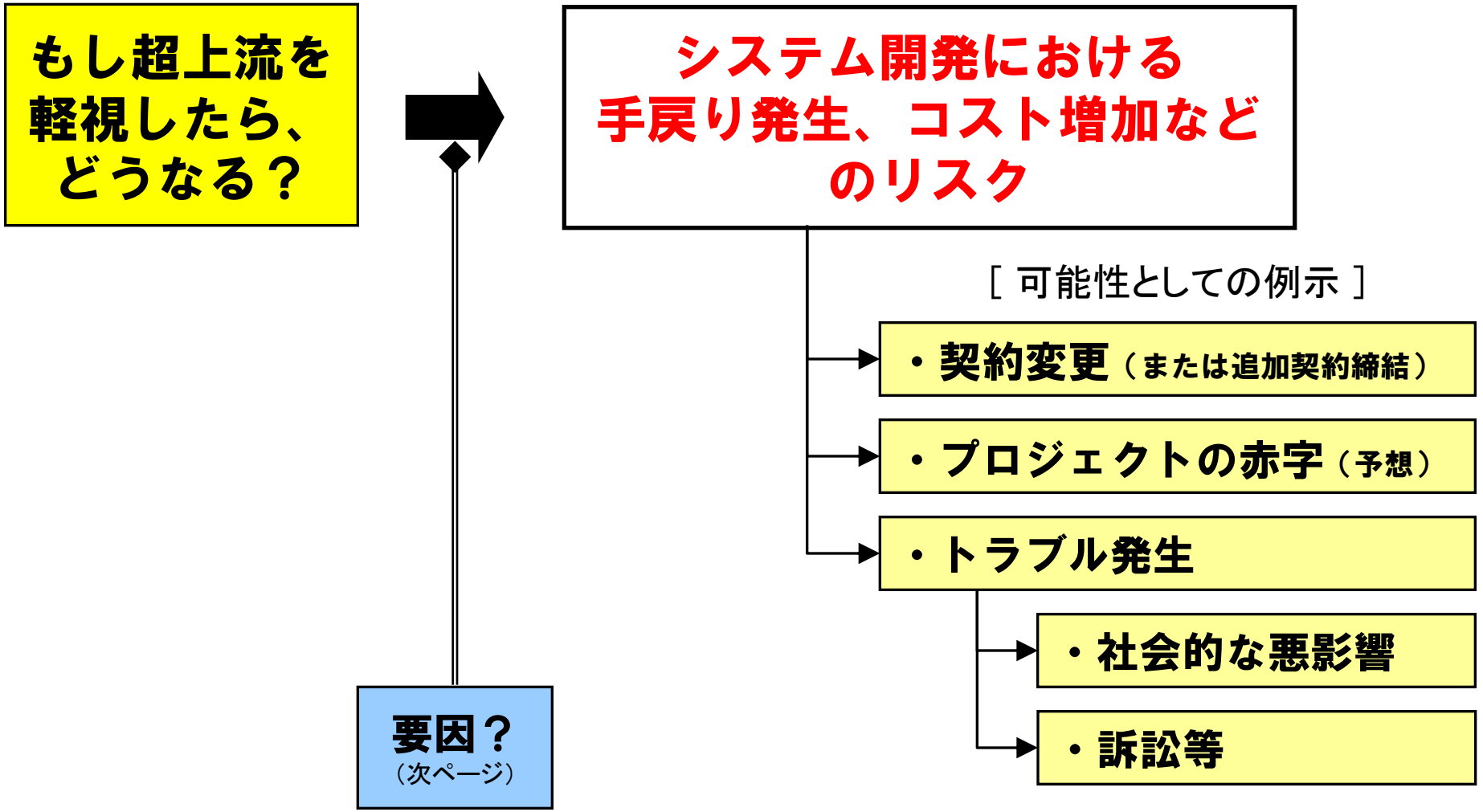
② 経営者の参画を（経営者としての役割があると）説いている。

③ 原理原則17ヶ条の活用による問題解決を提唱している。

3. 超上流とは？



4. もし超上流を軽視したら？



5. 共通フレームに含まれている主な考え方

（１）「利害関係者の役割と責任分担の明確化」を提唱

（２）「多段階の見積り方式」を提唱

（３）「Ｖ字モデルの採用」を提唱

（４）「超上流における準委任契約の採用」を提唱

（５）「要件の合意及び変更ルールの事前確立」を提唱

（６）「非機能要件の重要性を認識すること」を提唱

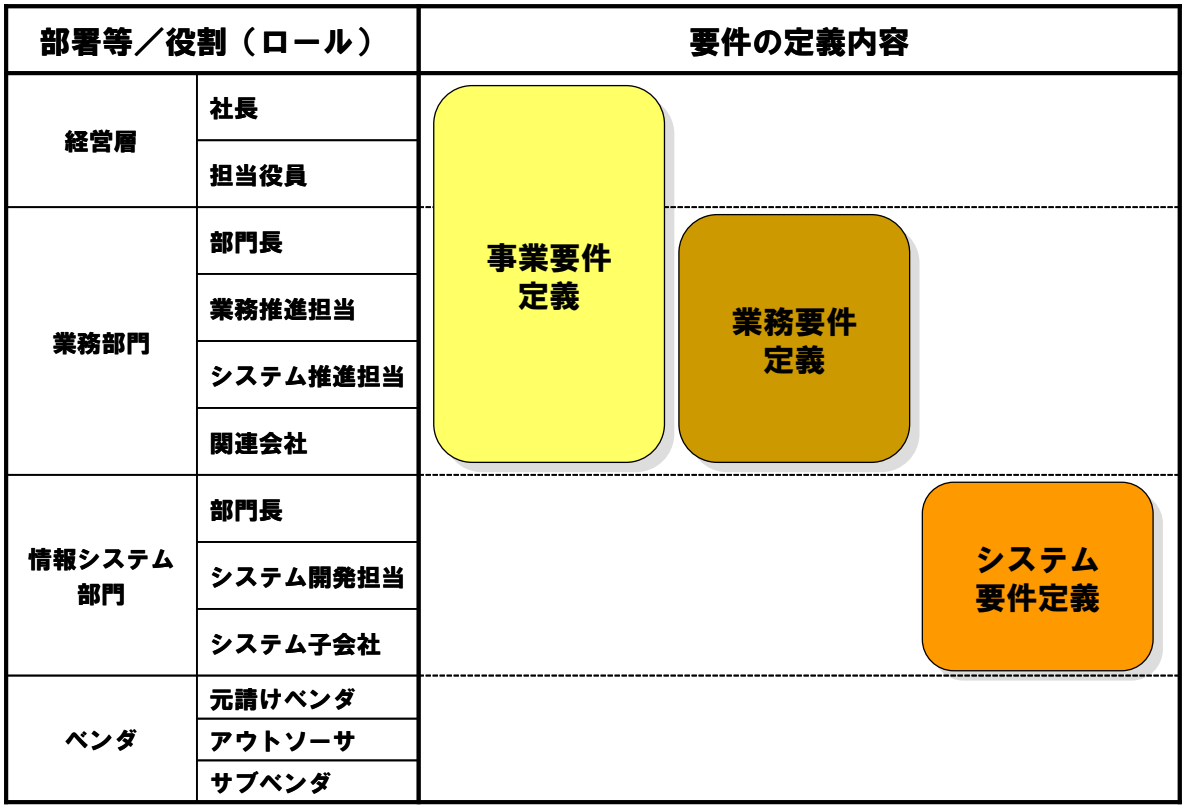
（７）「運用・保守を含めたSLCPを考えること」を提唱

（８）その他重要項目

（１）「利害関係者の役割と責任分担の明確化」を提唱

【参照先】 『超上流の本』:p.37 の 3.2（1）項、p.41 の 4.1

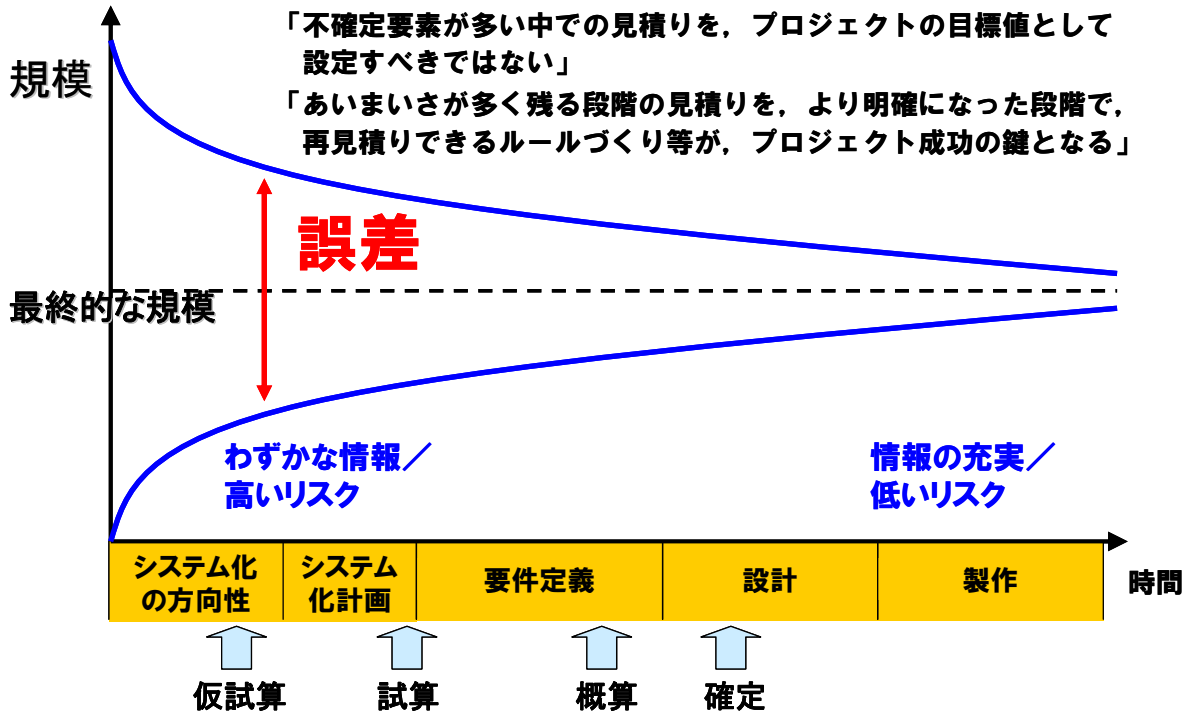
事業要件、業務要件、システム要件を定義できるのは、それぞれ経営層、業務部門、情報システム部門である。それぞれが責任をもって自らの役割を果たすことで、要件を適切に定義できる。



（2）「多段階の見積り方式」を提唱

【参照先】 『超上流の本』:p.38 の 3.2(2)項、原理原則15

わずかな情報で見積ること自体、リスクが高い。それ故、それだけで、プロジェクトの目標としてはならない。

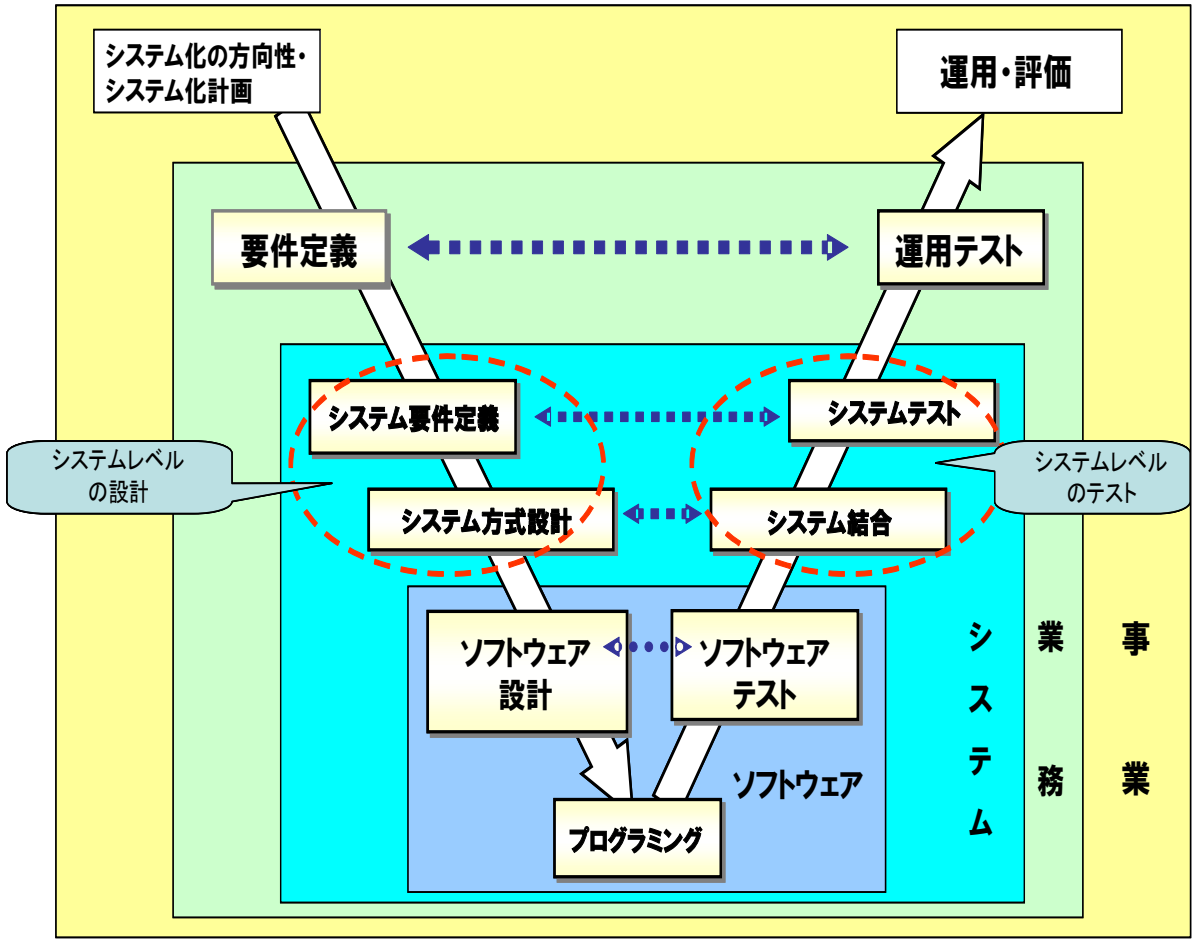


※SEC BOOKS「経営者が参画する要求品質の確保 ～超上流から攻めるIT化の勘どころ～（第2版）」より引用・一部改修

（3）「V字モデルの採用」を提唱

【参照先】 『超上流の本』:p.24 の 図2.3

設計（品質の埋め込みプロセス）とテスト（品質の検証プロセス）とを対応させることにより、プロダクト品質を確保する。



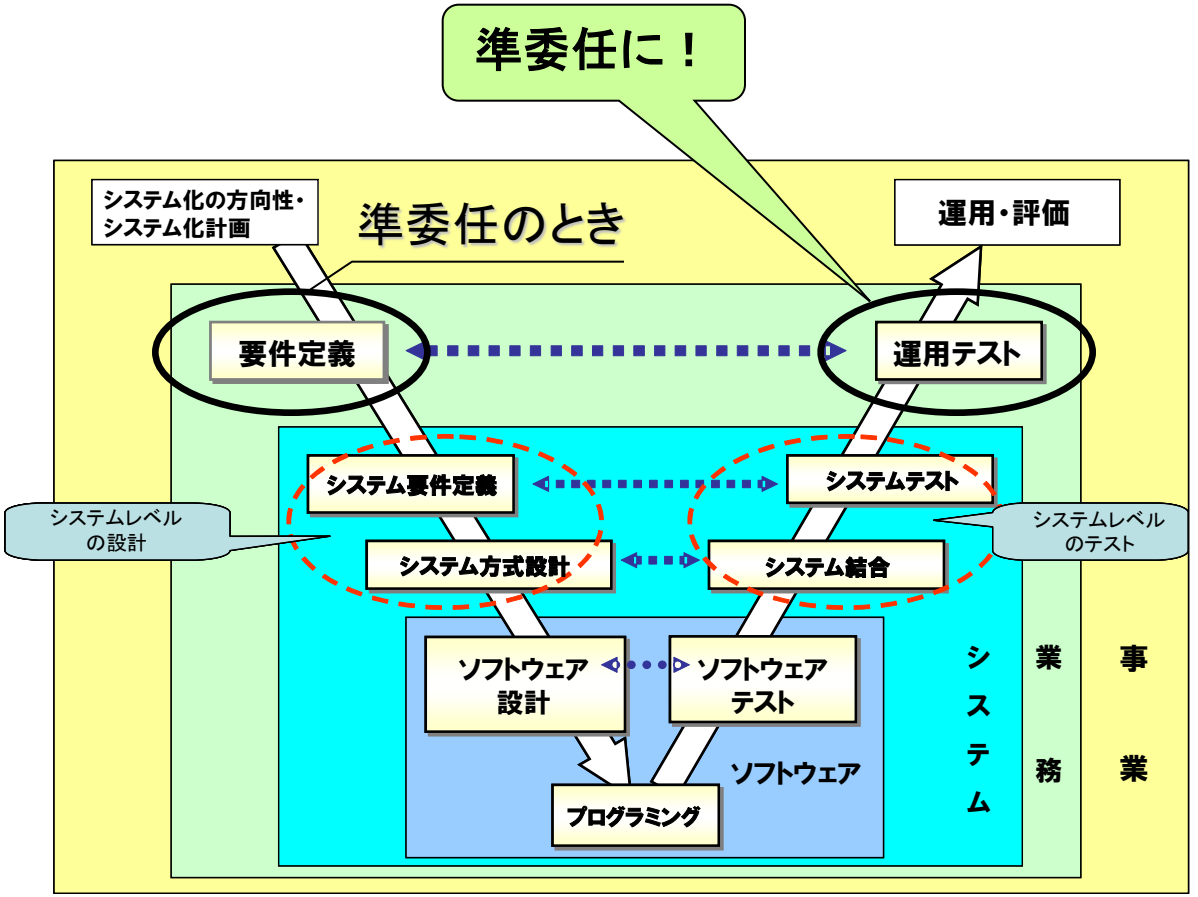
（４）「超上流における準委任契約の採用」を提唱

【参照先】 経済産業省「情報システムの信頼性向上のための取引慣行・契約に関する研究会」報告書
”～情報システム・モデル取引・契約書～”（ 2007年4月13日 公表 ）

超上流は、基本的には、ユーザ責任であるため、ベンダにとって準委任契約とするのが合理的である。（もし請負契約にすると、ユーザの事情に大きく影響されるため、リスクが大きい）。

【例】

- **超上流　→　準委任ならば**
  運用テスト → 準委任 に
- **ソフトウェア開発 → 請負**



（５）「要件の合意及び変更ルールの事前確立」を提唱

【参照先】 『超上流の本』:p.39 の 3.2（4）項

ソフトウェア開発においては、時の経過に伴って「要件は変わるもの」であり、ユーザとベンダとが事前にルールを策定し合意（確定）しておかないと、いざトラブルが発生した時に、速やかな対応が取れない。

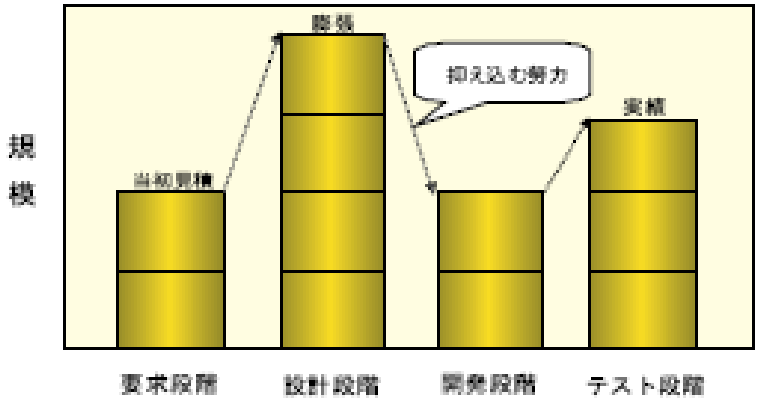


図 2.9　2-4-2-3の法則

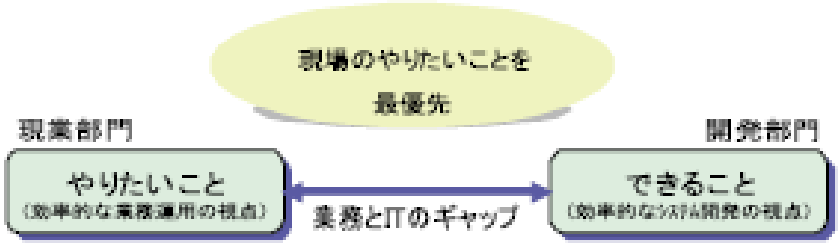


図 2.10　規模が膨張する力学

【出所】『超上流の本』p.31 より。

（６）「非機能要件の重要性を認識すること」を提唱

【参照先】 『超上流の本』:p.11 の 1.3（3）②項、p.39 の 3.2（3）項

運用テストの段階に至って、問題をもたらす要因は、機能要件のみならず、むしろ深刻な事態になりがちな非機能要件の方であるため、早い段階で「非機能要件の重要性」を認識し、何かしらの対応策を講じることが望ましい。

●機能要件　とは

システムに実装する機能に関する要件のこと。

●非機能要件　とは

運用要件、移行要件、性能要件、セキュリティ、機密情報保護対策など、機能要件以外の要件のこと。

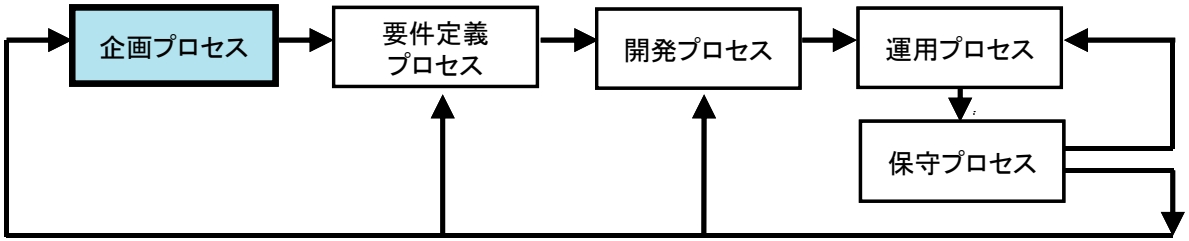
【出所】『超上流の本』p.43 より。

【注意】 業務部門（システムの利用部門等）にとっては、業務要件こそが重要である。
なお、業務要件に、機能要件、非機能要件も含まれる。
（業務要件については、『共通フレーム』（第2版）:p.112 の第3部 1.5.2.2 参照）

（７）「運用・保守を含めたSLCPを考えること」を提唱

【参照先】 『超上流の本』:原理原則6、7

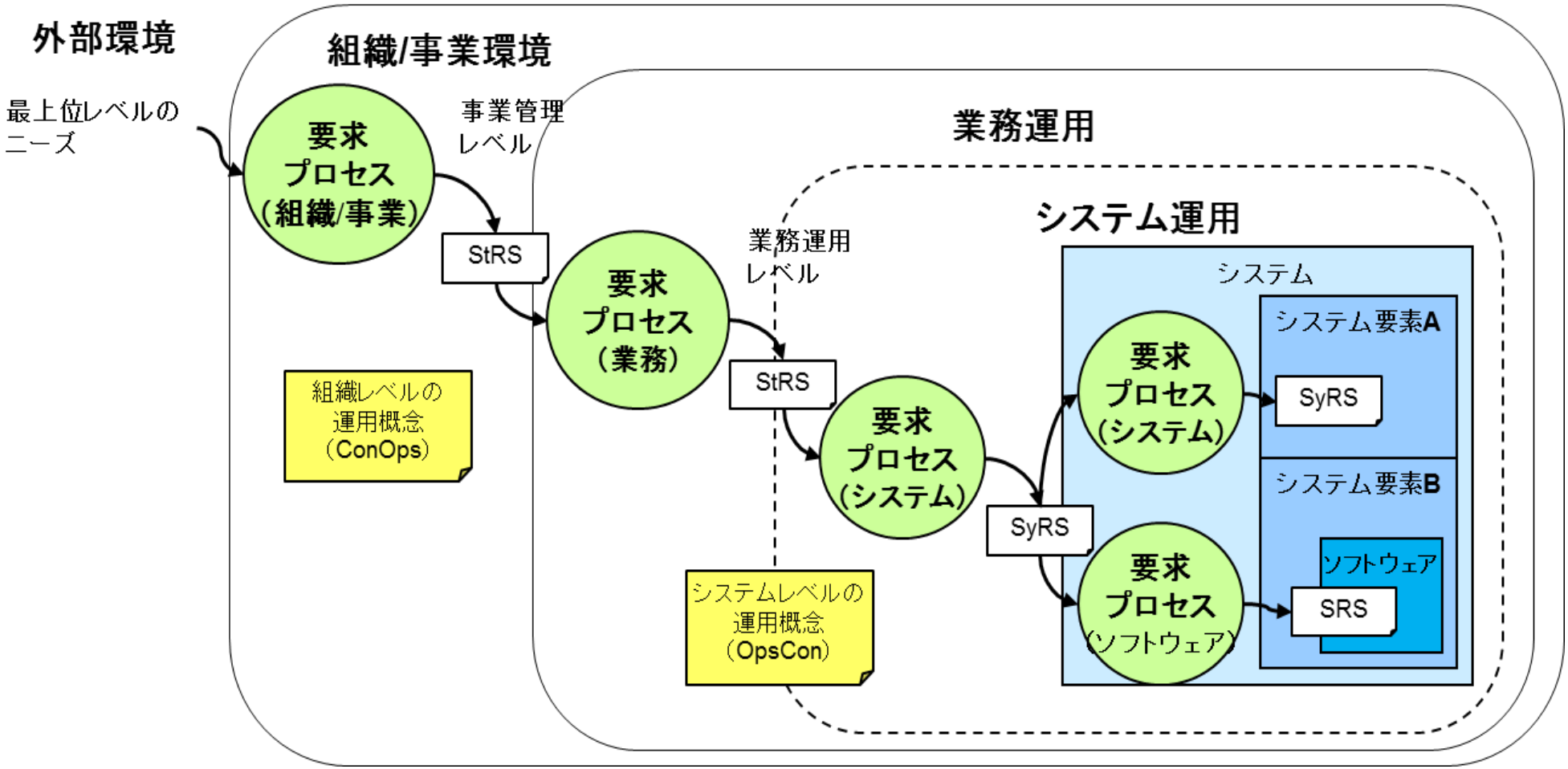
システムは生きもの。作って終わりではない。顧客との取引が継続する限り、または事業や業務が続く限り（ITシステムを必要とする限り）、システムライフサイクル全般に目配せしてシステム化計画（企画）や要件定義を行うことが、結局は、適正コストで「使えるシステム」を実現できる。



（８）その他重要項目

①要件の４階層（ISO/IEC/IEEE 29148より）
②「保守プロセス」の規定内容を解説
③Ｖ＆Ｖの適用場面の解説
④ソフトウェア保守規格の関連情報を紹介
⑤「４つの保守タイプ」を紹介

①「要件」の４階層（ISO/IEC/IEEE 29148より）



StRS：ステークホルダ要求　SRS：ソフトウェア要求

SyRS：システム要求

ISO/IEC29148　JIS原案より

②「保守プロセス」の規定内容を解説

【背景】保守プロセスから呼出すプロセスとして「開発」以外にも、「企画」「要件定義」「運用」プロセスもあることを明記する。

【訂正】（第2版の p.146で、以下のように下線部分を挿入）

「 保守者は，このプロセスを管理するために具体化した管理プロセス（3.1参照）に従って，保守プロセスをプロジェクトレベルで管理する。環境整備プロセス（3.2参照）に従って，**保守プロセスの基盤となる環境を確立する。修整プロセスに従って，**保守プロセスをプロジェクトに適した形に修整する。 ～ 」

【追加】（第2版の p.150、p.151）

「1.8.2.4 文書化

～

また，文書化し選択した修正案は，検証プロセス（2.4参照）又は妥当性確認プロセス（2.5参照）に従って検証又は妥当性確認を実施する。」

「1.8.3.2 修正の実施

～

また，修正対象によっては，企画プロセス（1.4参照），要件定義プロセス（1.5参照），運用プロセス（1.7参照）を利用してもよい。」

第4部の「**2.2 主プロセスから主プロセスの呼出し方とその関係**」において、この規定部分を引用している。

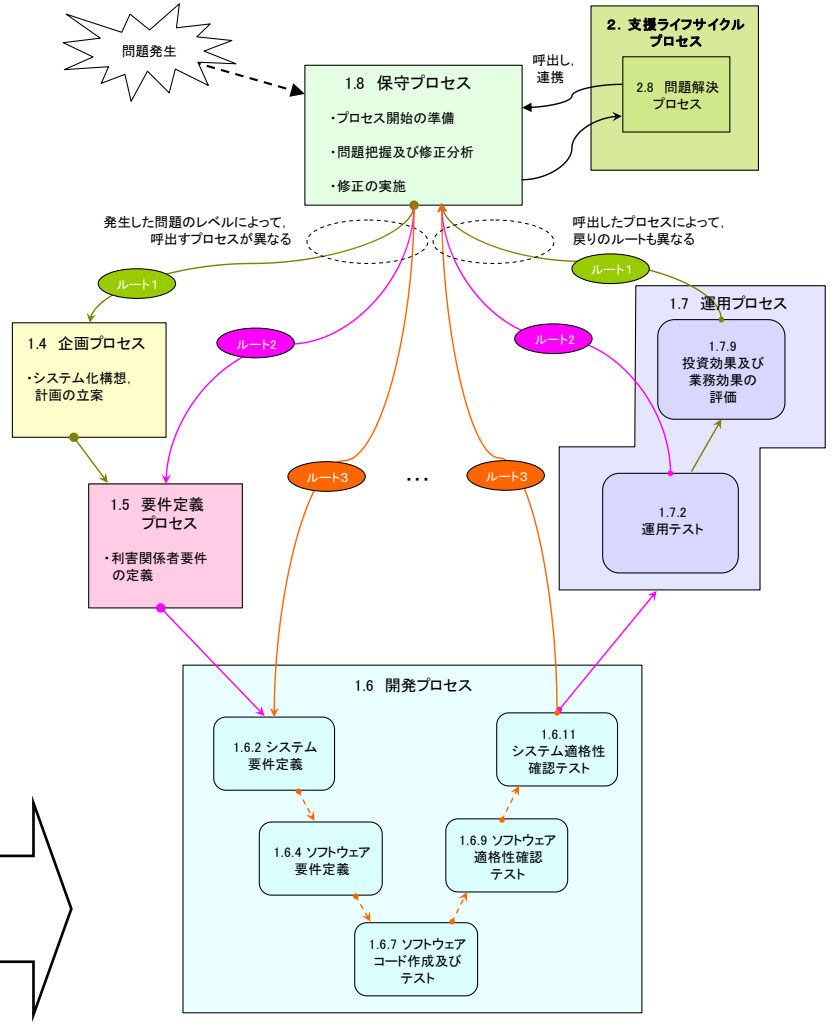


図4-23 主プロセスから主プロセスの呼出し例

③V&Vの適用場面の解説

【背景】V&V（Verification and Validation）の適用場面を示し、何と何とを比較するのかを明確にし、読者の方々に理解・実践してもらうことにより、システム及びソフトウェアの品質向上を図る。

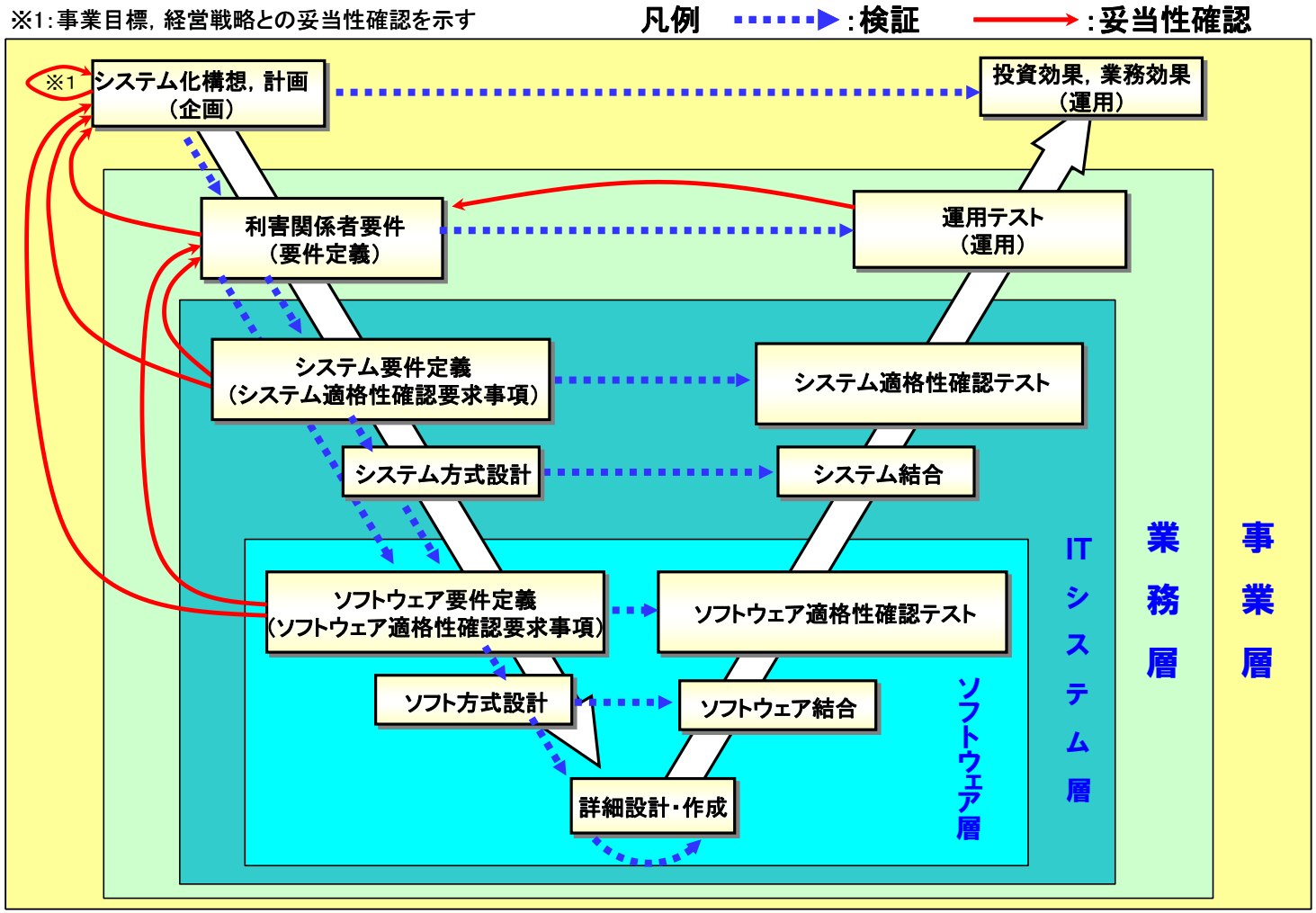


図4-xx　検証（Verification）と妥当性確認（Validation）の適用場面

④ソフトウェア保守規格の関連情報を紹介

【背景】ISO/IEC 14764（ソフトウェア保守）が1999年版から2006年版に改訂されたことに伴い（対応するJIS X 0161は2002年版→2008年版へ）、その改訂内容を共通フレーム2007に取り込むとともに、規格情報をも提供する。

【補足説明集 2.3 の目次構成】

2.3 保守プロセスに関連する情報

（1）ソフトウェア保守規格について

（a）ソフトウェア保守規格 ISO/IEC 14764 （JIS X 0161）の概要

・・・

＜ISO/IEC 14764（JIS X 0161）の位置付け、特徴、保守プロセスから関連するプロセスを呼出す場合の参照先を解説している＞

（b）規格にみる４つの保守タイプ

・・・

＜是正保守／予防保守／適応保守／完全化保守の定義を示している＞

（2）ソフトウェア保守に関する課題と対応

（a）ソフトウェア保守プロセスの改善は喫緊の課題

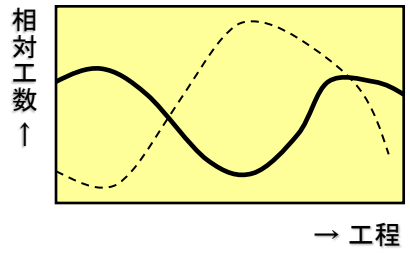
・・・

＜ソフトウェア保守の重要性を説くとともに、新規開発との相違点を解説している＞

（b）ソフトウェア保守のプロセス改善に向け「ふたこぶラクダ」の特性を知る

・・・

＜「ふたこぶラクダ」の特性について解説している ＞



（※） 補足説明集 2.3 の改訂に際して、次の方のご協力を頂きました。ここに、御礼を申し上げます。

増井 和也 様（東芝ソリューション株式会社）

（ソフトウェア・メインテナンス研究会（SERC）の幹事のおひとりで、JIS X 0161 原案作成委員会にも関わっておられます。

SERC（Software Evolution Research Consortium）：ソフトウェアのメインテナンスを専門に研究する非営利任意団体。）

⑤「４つの保守タイプ」を紹介

【背景】ソフトウェア保守といっても、様々なタイプがあることを紹介する。なお、実際には複数の保守タイプが混合する案件も多い。このようなソフトウェア保守の多様性に対応して、保守プロセスの最適化（修整）が求められる。

【 ① 是正保守（corrective maintenance） 】

ソフトウェア製品の引渡し後に発見された問題を訂正するために行う受身の修正（reactive maintenance）。

（注記）この修正によって，要求事項を満たすようにソフトウェア製品を修復する。
なお，是正保守の一部として，是正保守実施までシステム運用を確保するための，計画外で一時的な修正として，「緊急保守（emergency maintenance）」がある。

【 ② 予防保守（preventive maintenance） 】

引渡し後のソフトウェア製品の潜在的な障害が運用障害になる前に発見し，是正を行うための修正。

「*引渡し後のソフトウェア製品の潜在的な障害が，故障として現れる前に，検出し訂正するための修正。*」と定義している。

【 ③ 適応保守（adaptive maintenance） 】

引渡し後，変化した又は変化している環境において，ソフトウェア製品を使用できるように保ち続けるために実施するソフトウェア製品の修正。

（注記）適応保守は，必須運用ソフトウェア製品の運用環境変化に順応するために必要な改良を提供する。これらの変更は，環境の変化に歩調を合わせて実施する必要がある。例えば，オペレーティングシステムの更新が必要になったとき，新オペレーティングシステムに適応するためには，幾つかの変更が必要かもしれない。これは，アプリケーションの全体機能要件は変わらないにも関わらず，オペレーティングシステムやミドルウェアの変更，ハードウェアの変更，法改正などに伴ってアプリケーションソフトウェアに影響する部分の改良が必要になるようなケースである。

【 ④ 完全化保守（perfective maintenance） 】

引渡し後のソフトウェア製品の性能又は保守性を改善するための修正(1)。

（注記）完全化保守は，利用者のための改良（改善），プログラム文書の改善を提供し，ソフトウェアの性能強化，保守性などのソフトウェア属性の改善に向けての再コーディングを提供する。業務の改善に伴う機能要件が変わるときに行う改良などを指す。

(1) この定義は，ISO/IEC 14764:1999（JIS X 0161:2002）から引用している。ISO/IEC 14764:2006（JIS X 0161:2008）においては，完全化保守と予防保守の定義が類似した表現となっているため，読者が混乱しないよう，あえて旧規格の定義を掲載した。

第2部　終わり

第３部　実務に活かすＩＴ化の原理原則１７ヶ条

SEC
Software Engineering for Mo·No·Zu·Ku·Ri

■ 超上流の重要なポイントを短い言葉でまとめ、原理原則17ヶ条とした。

原理原則【1】	ユーザとベンダの想いは相反する
原理原則【2】	取り決めは合意と承認によって成り立つ
原理原則【3】	プロジェクトの成否を左右する要件確定の先送りは厳禁である
原理原則【4】	ステークホルダ間の合意を得ないまま、次工程に入らない
原理原則【5】	多段階の見積りは双方のリスクを低減する
原理原則【6】	システム化実現の費用はソフトウェア開発だけではない
原理原則【7】	ライフサイクルコストを重視する
原理原則【8】	システム化方針・狙いの周知徹底が成功の鍵となる
原理原則【9】	要件定義は発注者の責任である
原理原則【10】	要件定義書はバイブルであり、事あらばここへ立ち返るもの
原理原則【11】	優れた要件定義書とはシステム開発を精緻にあらわしたもの
原理原則【12】	表現されない要件はシステムとして実現されない
原理原則【13】	数値化されない要件は人によって基準が異なる
原理原則【14】	「今と同じ」という要件定義はありえない
原理原則【15】	要件定義は「使える」業務システムを定義すること
原理原則【16】	機能要求は膨張する。コスト、納期が抑制する
原理原則【17】	要件定義は説明責任を伴う

原理原則17ヶ条の構成

SEC
Software Engineering for Mo·No·Zu·Ku·Ri

■ **原理原則条項：**

原理原則は「超上流」において必要とされる事柄を、格言のように短く表現したもの

■ **基本的な考え方：**

原理原則を理解しやすくするため、原理原則の基になる考え方を説明したもの

■ **行動規範：**

原理原則の基づいて、受注者・発注者のそれぞれが具体的にどのように行動すべきか示したもの

原理原則[1]

ユーザとベンダの想いは相反する

ITシステムの企画・開発の現場では、ユーザ企業とベンダ企業の相反する想いがあります。例えば、ユーザ企業は、要件はできるだけじっくり詰めたいし、予算は早期の投資判断を求められるので最終費用を早く確定してほしいとの想いがあります。他方のベンダ企業の想いはまったくその逆です。これがお互いにとってそもそもの不幸の始まりとなります

開発規模（工数）に見合った、最低限の工期を確保できなければ顧客満足を満たす開発はできません。受注者には開発規模に見合った工期を主張することが求められます。

ユーザ企業・ベンダ企業の相反する想い

SEC-BOOKS経営者が参画する要求品質の確保』より

原理原則[2]

取り決めは合意と承認によって成り立つ

証拠のない口約束のように、決まったと了解していることが、それ以降の都合で無責任に変更となり、残念な思いをする、ということはよくあります。
決め事は可能な限り文章に残し、承認ルール（主体と方法）の確認をして、信頼度を高めなければいけません。
承認は合意に基づいていることが必要です。

原理原則[3]

プロジェクトの成否を左右する要件確定の先送りは厳禁である

**要件定義は開発全体の成否を左右重要な工程です。
曖昧な要件のまま開発が始まると、プロジェクトが失敗するリスクが大きくなります。
特に、システムの出来を左右する要件に高いリスクを抱えたまま、プロジェクトを進めることは危険です。
あせってベンダに開発を依頼しても、先に進めず、かえって時間・コストがムダになることもあります。
解決の目処が立つまでは、先に進まない勇気も必要です。**

原理原則[4]

ステークホルダ間の合意を得ないまま、次工程に入らない

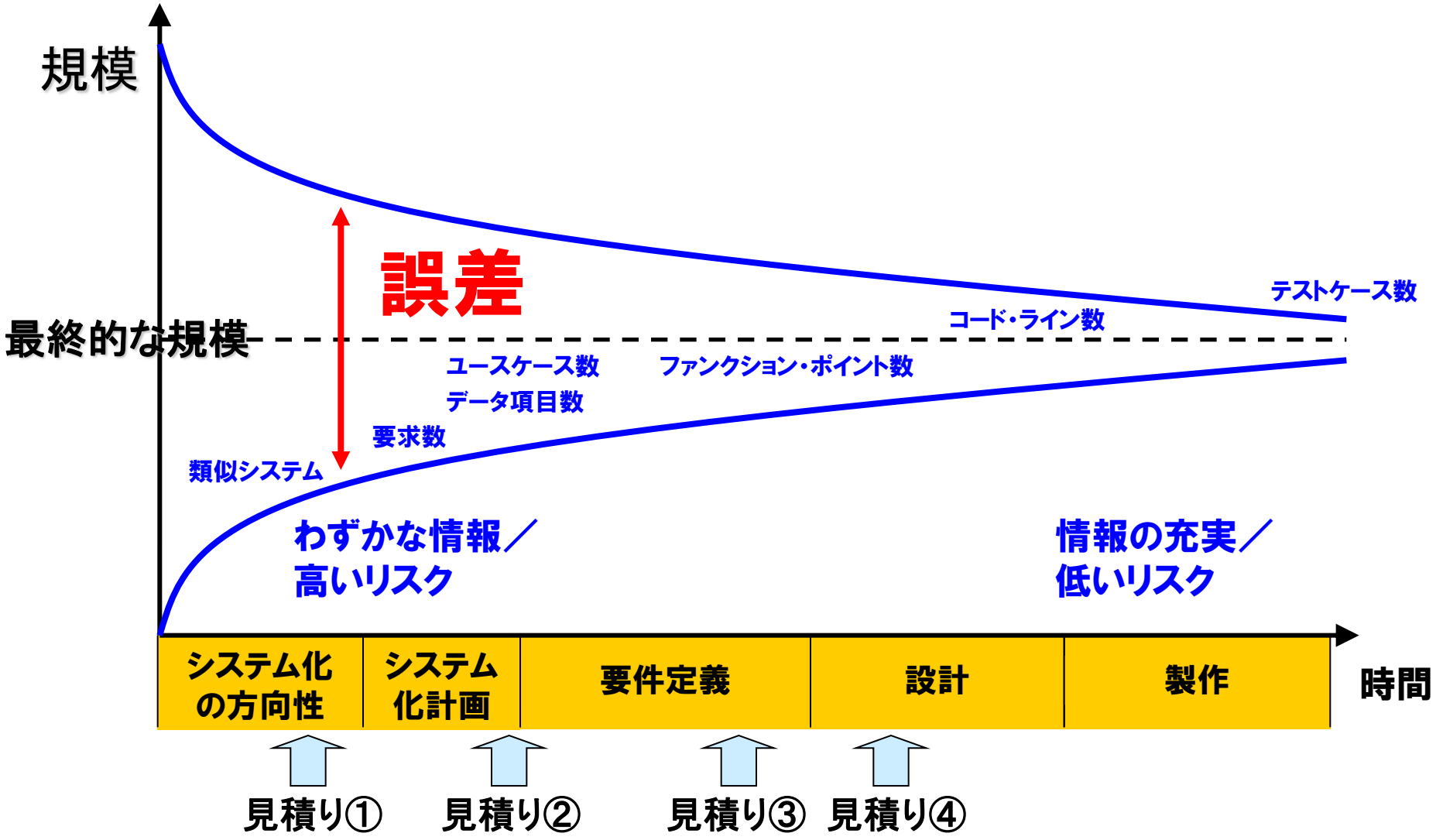
プロジェクトを起こした業務企画担当者は、プロジェクト責任者として、これらステークホルダの方針、意見、課題などについて、漏れなく綿密に把捉し、できることとできないことをIT担当者、ベンダとともに切り分け、業務要件として取りまとめていく責任を果たす必要があります。
ステークホルダもまた、システムの供給側に立つ場合は、積極的にシステム開発要件の策定に参加し、利用者ニーズを確実に把握して、正確にシステム機能に反映していくことが必要です。

原理原則[5]

多段階の見積りは双方のリスクを低減する

不確定要素が多い中での見積りをプロジェクトの目標値として設定すべきではありません。
あいまいさがある段階の見積りを、はっきりした段階で見積り直せるルールづくりなどがプロジェクト成功の鍵となります。
要件の不確定さやプロジュクトの特性・リスクに応じて、適切な契約方式（多段階契約、インセンティブ付契約など）を選択することにより、発注者・受注者の双方にメリットが生まれます。多段階とは、受注先をその都度変えるということではなく、固まり具合に応じて見積り精度をあげていこうということです

見積り時期とリスク



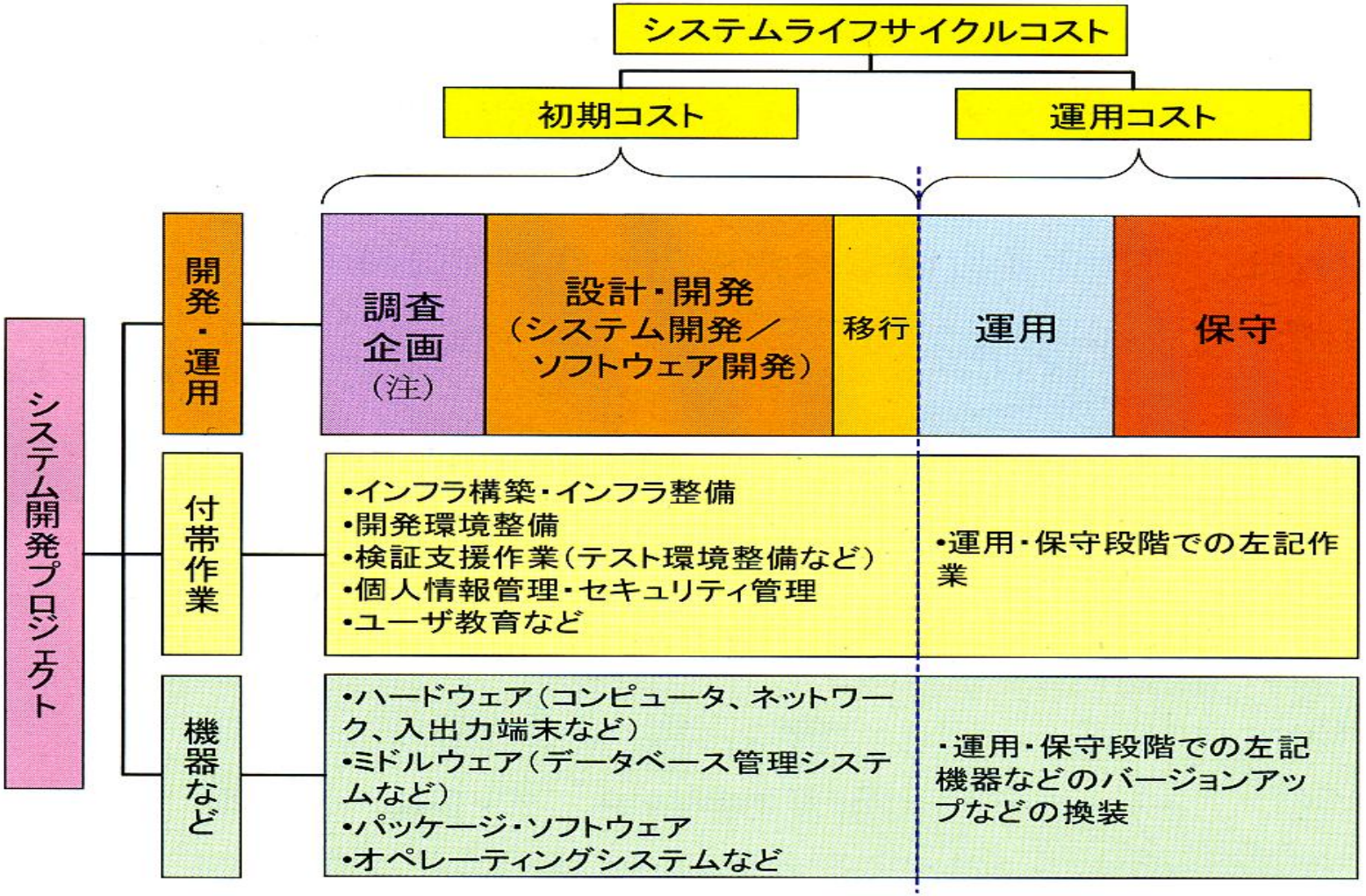
（注）文献：Barry Boehm 著の“Software Engineering Economics（Prentice-Hall社）”の図に基づきSEC作成

原理原則[6]

システム化実現の費用はソフトウェア開発だけではない

見積り範囲がソフトウェア開発のことだけを指しているのか、インフラ整備（システム基盤整備）などのような付帯作業も対象にしているかなど、スコープを明確にしていくことが大切です。
発注者は、何をお願いし、何を自分で行うのか、一方、受注者は自分の提供する作業やサービスはどの範囲なのかをお互いに明確にしておくことが重要です。

システム開発プロジェクトの構成要素



（注） システム化の方向性から要件定義

原理原則[7]

ライフサイクルコストを重視する

開発コスト、運用・保守コストのバランスを考えなければなりません。大切なことはライフサイクルコストを意識することです。
例えば、運用性・保守性を高めるポイントとして以下があります。
ーメンテナンスフリー
ー拡張性の容易さ確保
ーモニタリング・トレーサビリティの確保
ー障害発生時の調査、リカバリーが容易な設計
ーOS・ハードウェアのバージョンアップ対応

原理原則[8]

システム化の方針・狙いの周知徹底が成功の鍵となる

超上流のフェーズで、システム化の方針・狙いを浸透させておかないと、各人が勝手気ままに要件を考えるため、仕様の統一に時間がかかり、最初の構築だけでなく、その後の維持・保守においても費用と時間が増大することになります。
システム化の目的はコンピュータやプログラムではなく、事業目標を達成するための情報システムの構築なのです。

原理原則[9]

要件定義は発注者の責任である

要件定義とは、どのようなシステム、何ができるシステムを作りたいのかを定義することです。それはあくまでも発注者の仕事であり、発注者の責任で行うものです。
要件定義があいまいであったり、検討不足のまま、受注者に開発を依頼した場合、その結果として、コスト増、納期遅れ、品質低下を発生させるおそれがあります。
その責任を受注者に負わせることはできません。
受注者が支援する場合であっても、要件定義で作成した成果物に対する責任は発注者にあります。

原理原則[10]

要件定義書はバイブルであり、事あらばここへ立ち返るもの

ベンダ企業を含むステークホルダ間の合意のベースとなるのは常に要件定義書です。設計工程以降よりも、むしろ、要件定義の合意形成時点での吟味が重要です。「決定先送り型」の要件定義では、あいまいな海図に基づく航海のようなもので、早晩プロジェクトが破綻します。
ステークホルダ間の合意は、名目的な合意ではなく、実質的な合意であることが不可欠です。

原理原則[11]

優れた要件定義書とはシステム開発を精緻にあらわしたもの

要件定義工程では、業務要件を整理・把握し、その実現のためのシステム機能要件をしっかり固めます。あわせて性能、信頼性、セキュリティ、移行・運用方法などの非機能要件、既存システム接続要件、プロジェクト特有の制約条件も洗い出します。また、将来の方針を見込んで稼働環境を定めることが大切です。流行に流されず、ルールを定めることです。

原理原則[12]

表現されない要件はシステムとして実現されない

この原則は、建築における施工主と工事業者の関係にあるように、発注と受注における常識です。しかし、情報システム開発においては往々にしてこの原則が成立しない場合があり、「行間を読め」、「言わなくても常識」、「言った言わない」など表現されない要件が、両者のトラブルの原因になります。

原理原則[13]

数値化されない要件は人によって基準が異なる

要件定義では、定量化できるものは、極力、数値化します。数えられないものは定義できません。「大きい、小さい、速い」だけでは、人によって「ものさし」が異なります。
数値化されていても誤りはあります。例えば、使用する単位が違えば結果は大きく変わります。単位まで含めて確認し、決めなければなりません。

原理原則[14]

「今と同じ」という要件定義はありえない

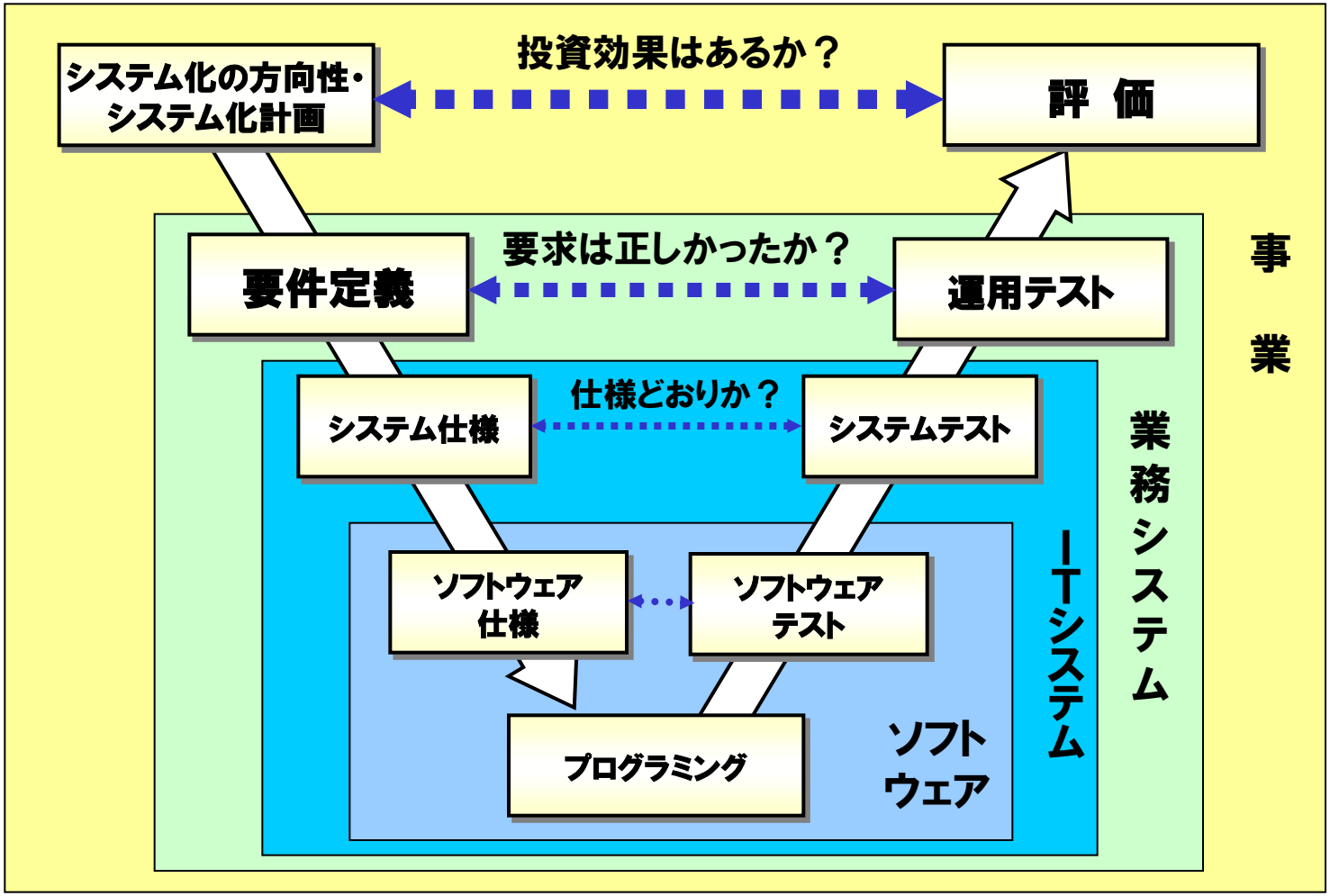
「今と同じ」でも要件定義は必要です。
そもそも同じでよいなら再構築する必要はありません。
よくないから再構築するというところから発想したいものです。
現行システムの調査をする場合は、システムの機能を洗い上げ、新システムの実像を明確にするだけでは不十分です。現行システムをどう使っているか、という点から調査をしなければなりません。
「そもそも今の要件はどうなっているのか」を問い直し、場合によっては具体的な要件にまで導くことも必要です。

原理原則[15]

要件定義は「使える」業務システムを定義すること

要件定義は、業務にとって「使える」、「役に立つ」、「運用できる」システムを定義することです。
発注者は、それまでのやり方にとらわれることなく、むだな業務や非効率な手順を客観的に評価し、新業務をゼロベースで再設計することが大切です。
要件定義の場に参加して、議論が横道にそれたり、枝葉末節に陥らないように助言するのは受注者の役割です。また、受注者は、要件として定義したものが、システム化計画で想定したコストや期間と比べて過剰なものや、逆にあまりに多くの費用を要さずとも実現可能な要件は勇気を持って変更を進言しなくてはなりません。

要件定義・仕様とテストの関係



原理原則[16]

機能要求は膨張する。コスト、納期が抑制する

システム開発のコストは実現する機能ではなく、工数に比例しますから、どのくらいの作業が残っているのかをきちんと把握しながら、機能との折り合いをつけて作業を進める必要があります。このバランス感覚をプロジェクトメンバー全員が持っていなければ意味がありません。プロジェクトの背景や目的に応じたシステム化の範囲を検討し、「ついでにこの範囲も」という考え方は本来の目的を見失うので絶対に避けましょう。

原理原則[17]

要件定義は説明責任を伴う

システム開発における万全なる準備は、正確な要件という情報の次工程に向けての伝達です。自分が次工程に伝える必要のある情報について、要件確定責任だけでなく説明責任を負う必要があります。
システム開発の受託側から見た原則は「受託した要件として、書いてあるものは実現させる。書かれていないものは作らない。」ことです。
もちろん、プロジェクトのスタート地点で、すべてを誤りなく責任をもって確定することはできません。「要件の行間を読め」ということを要求してはいけません。
基本的には当たりまえの前提や例外処理であっても漏れなく伝達する必要があります。

ご清聴ありがとうございました