Legal Engineering for Software-Defined Society（LE4SDS）における構造可視化手法の検討

2026 年 3 月

独立行政法人情報処理推進機構
Information-technology Promotion Agency, Japan

目次

1. 背景と課題意識

1.1 事業環境の変化と AI 時代の制度ギャップ

データ利活用やデジタル化が進むにつれ、組織は外部委託・業務提携・クラウド利用・サプライチェーン連携などを通じて、組織境界を越えた情報流通を前提に事業を設計するようになっている。こうした環境では、従来の「境界を守れば安全」「社内は基本的に信頼できる」といった前提が揺らぎ、利活用の促進とリスク低減の両立が重要な経営課題となる。

加えて、AI 時代の到来により、技術変化と社会実装のスピードが加速し、制度・ガイドライン・運用慣行といった「ルール側の更新」が相対的に追いつきにくくなっている。生成 AI や高度な分析は、従来は想定されにくかった推論・結合・再利用を容易にし、同じデータであっても用途や影響範囲が短期間で変化し得る。結果として、過去の想定で整備された手続や統制が、現状のリスクに対して過剰にも過小にもなりやすいという課題が顕在化している。

1.2 Legal Engineering for Software-Defined Society（LE4SDS）の必要性

今後、社会の多くの機能がソフトウェアによって定義・更新される「Software-Defined Society」へ移行するにつれ、法令・契約・社内規程といった規律を、ソフトウェアの設計・実装・運用と切り離して扱うことは一層難しくなる。求められるのは、規律を単に遵守事項として「読む」のではなく、設計要件・統制・運用プロセスとして「作り込み」、変更に追随できる形で維持することである。

本書で扱う **LE4SDS（Legal Engineering for Software-Defined Society）** は、SDS（Software-Defined Society）を実現するために、法・標準等のルールを要求として扱い、設計・実装・運用・評価・更新までを循環させるリーガルエンジニアリングの方法論である。本書では、この方法論を実務に落とし込むための「構造可視化手法」として、Step1〜Step3 および 3 つの View（View A/B/C）を提示し、具体例を用いて検討する。重要なのは、制度やルールの「正しさ」を論じるだけでなく、「なぜ必要か」「どのリスクに対してどの要求強度が必要か」「運用で維持できるか」を関係者間で共有できる形に落とすことである。

なお、法令は規範の一種であり、外部強制力や当局執行といった属性を持つ点で特徴的であるが、本書が扱う構造化の骨格は規範一般に共通する。

1.3 本書の課題設定とアプローチ（構造化・可視化）

一方で、実務の議論は個別論点に分断されやすい。例えば、法務は規律や手続、企画は価値・収益、技術は実装・運用、監査は証跡・統制といったように、関心の粒度や語彙が異なるため、同じ対

象を議論していても「何を守り、何を達成し、どの手段で担保するのか」が揃わないことがある。その結果、過剰な統制による機会損失、または統制不備による事故・不祥事といった不整合が生じやすい。

この課題に対し本書は、法令に限らず、制度・規程・契約・ガイドライン等を含む「規範（Normative System）」を対象として、LE4SDS の考え方を実務に適用するための前提となる「構造」を提示する。具体的には、リスク起点・Outcome 志向で意思決定の筋道を保ちつつ、要求、統制、実装、運用上の実効性（保証）を一貫して接続する枠組みを、次の 3 ステップとして整理する。

- **Step1：規範が果たすべき機能の特定（目的・保護対象・要求・Outcome の明確化）**
  規範が守ろうとする保護対象（法益など）や目的を起点に、規範が果たすべき機能を整理し、達成すべき状態（Outcome）と要求（Requirement）として合意可能な形に構造化する。
- **Step2：機能を実現する統制の特定（統制サービスと実装手段の設計）**
  Step1 の要求・Outcome を実現するために、どの統制（Control）をどこに配置するかを設計し、技術・運用・組織措置等の実装手段（Mechanism）へ落とし込む。
- **Step3：実効性と説明可能性を担保する保証・更新設計（運用保証ループ）**
  統制が「導入して終わり」にならないよう、評価→監視→是正→開示→監査の循環により有効性を継続的に確認・改善し、環境変化や新たな知見に応じて要求・統制・証拠を更新するための運用構造を設計する。

上記 3 ステップを、本書では設計・説明・運用の観点から 3 つの View として可視化する。すなわち、(1)動機・リスク・原則・要求・Outcome を整理する **View A**、(2)統制を工程に配置し実装へ落とす **View B**、(3)評価・監視・是正・開示・監査の循環で実効性を維持し、更新を駆動する **View C** である。これらの View は規範の種類（法令／契約／社内規程等）に依存しない共通の骨格を提供し、規範ごとの差異は主として強制力、執行主体、制裁、適用範囲、更新頻度等の「規範の属性」に現れる。概ね、Step1 は View A、Step2 は View B、Step3 は View C で具体化される（実務では反復する）。

また、可視化の手段は複数あり得るが、本書では抽象度の異なる論点を一貫して接続しやすいという理由から、具体例として ArchiMate を採用する。あわせて、個人情報保護のユースケースを題材に、本方法論の適用可能性と読み取り方を具体化し、合意形成や設計判断の説明にどのように寄与し得るかを検討する。なお、個人情報保護は規範の代表例として扱うものであり、本手法は他の領域（例：安全、品質、AI ガバナンス等）にも同型に適用可能である。

2. 「構造」を可視化する意義

本章では、LE4SDS の考え方を「構造」として可視化することの意義を述べる。ここで用いる可視化手法は ArchiMate に限られず、目的や読者、既存の成果物に応じて、他のアーキテクチャ記述・モデル化手法、表形式のトレーサビリティ、文章テンプレート等も選択し得る。本書では、動機・リスクから統制・実装・運用上の実効性（保証）までを一貫して接続できる点を重視し、具体例として ArchiMate を採用する。

可視化の狙いは「図をきれいに描く」ことではなく、複数部門に跨る論点を、同じ語彙と粒度で扱えるようにし、意思決定の根拠と実装・運用を接続することである。とりわけ LE4SDS が重視するリスク起点・Outcome 志向の議論では、結論（統制や手続）だけを先に置くと、なぜその強度なのか、なぜ別案ではないのかが説明しにくい。可視化は、こうした「説明の筋道」を構造として残すための手段である。

実務では、特定の技術や手段（例：暗号化、匿名化、ゼロトラスト、同意取得の徹底等）を先に置き、「それで守れるはずだ」と発想してしまうことが少なくない。しかし、手段は目的ではなく、リスクと Outcome（達成したい状態）に照らして初めて適切な強度や組合せが決まる。手段先行の設計は、過剰統制による機会損失、または想定外のリスクの見落としにつながりやすい。本書が示すアプローチは、まずリスク起点で説明の筋道（Assessment→Principle→Requirement→統制・実装→保証）を確立し、その上で必要な技術や手続を選ぶことを重視する。

2.1 可視化は単なる「図示」ではなく「合意形成のための共通言語」

可視化の価値は、部門ごとに分断されがちな論点を、同じ「対象」として扱える点にある。法務は規律や手続、企画は価値や目的、技術は実装と運用、監査は証跡と統制を主に見るため、文章だけで議論すると、同じ話をしているつもりでも「前提」「守りたい対象」「担保の手段」が噛み合わないことがある。結果として、過剰統制（機会損失）か、統制不足（事故・不祥事）に振れやすい。

本書で具体例として採用する ArchiMate は、動機・評価（リスク）から要求、統制配置、実装、運用上の実効性（保証）までを、要素と関係のネットワークとして表現できる。これにより、例えば「なぜこの統制が必要か」をリスク・原則から辿れ、「この要求はどの統制で満たされるか」を実装まで辿れる。重要なのは、ArchiMate そのものというより、「抽象度の異なる論点を一貫して接続し、読み替え可能な共通言語に落とす」という点である。

なお、この効果は ArchiMate に固有のものではなく、表形式のトレーサビリティ、他のアーキテクチャ記述、文章テンプレート等でも一定程度狙い得る。本書では、3 章で述べる 3View（View

A/B/C）の分解と親和性が高く、因果とトレーサビリティを同居させやすい点から、説明の具体例として ArchiMate を用いる。

2.2 可視化がもたらす 3 つの効果

可視化の効果は大きく三つに整理できる。いずれも「結論の正しさ」ではなく、「結論に至る筋道が説明可能か」「運用で崩れないか」に効く。

(1) 説明責任の強化

統制や技術措置が「なぜ必要か」「なぜその強度か」を、リスク評価（Assessment）と原則（Principle）から辿れる。これにより、設計判断が属人的な経験則に留まらず、関係者や監査に対して再説明可能な形で残る。

(2) 抜け漏れ・二重化の早期発見

要求はあるが運用ループがない、統制はあるが実装への接続がない、といった欠落を発見しやすい。また、目的の異なる統制が同じ実装で二重に実現されている等の「重複」も見えやすくなる。

(3) 変更耐性の向上

新規ユースケース追加や法改正・技術更新などが発生した時に、どの要求・統制・運用に影響するかを構造として点検できる。結果として、変更のたびにゼロから議論をやり直すのではなく、影響範囲を限定して合意形成を進められる。

2.3 「これから作る」局面での価値

構造の可視化は、既に決まった制度や方針を後から整理する用途にとどまらない。むしろ、これから設計・更新する局面において、仮説モデルを置いて論点を固定し、関係者間の合意形成を前倒しする効果が大きい。文章で議論を積み上げた後に図にするのではなく、モデルを叩き台にして議論し、要素と関係を確定させながら文章化することで、設計空間の整理と意思決定を加速できる。

特に重要なのは、可視化が「後追いの整理」だけでなく、「設計の前倒し」に効く点である。3 章で述べるように、View A で動機・リスク・原則・要求を先に固定し、View B で工程と統制配置に落とし、View C で保証と更新を設計に内蔵する、という順序で整理すると、議論の未確定点を早期に顕在化できる。すなわち可視化は、合意形成を支える成果物であると同時に、合意形成そのものを進めるための「作業台」となる。

3. LE4SDS の考え方を 3 つの視点で捉える枠組み

LE4SDS の考え方を「構造」として保持し、部門横断で説明可能にするためには、同一の対象を単一の図で表し切ろうとせず、焦点の異なる複数の視点（View）で分解して表現することが有効である。単一 View にすべてを詰め込むと、線が増えすぎて読み解けなくなる一方、View を分けすぎると全体の整合やトレーサビリティが失われる。したがって、「少数の視点に固定し、同じ要素を共有しながら焦点を切り替える」設計が重要となる。

本書が対象とする「規範」は、法令に限らず、制度・規程・契約・ガイドライン等を含む。規範の種類によって、強制力、執行主体、制裁、適用範囲、更新頻度などの属性は異なるが、本書が示す構造化の骨格（動機・リスクから統制・運用保証までを接続する枠組み）は規範一般に共通する。

また、1 章で述べた 3 ステップ（Step1：機能と Outcome・要求の特定／Step2：統制と実装への落とし込み／Step3：保証・更新設計）は、概ね View A／View B／View C でそれぞれ具体化される（実務では反復する）。

本書では、(A)「なぜ／何を」（動機・評価・原則・要求）、(B)「どこで／何で」（対象の流れ・工程への統制配置と実装）、(C)「続けて守れるか」（評価・監視・是正・開示・監査の循環）という 3 つの視点を採用する。これにより、意思決定の根拠（リスク起点）から運用上の実効性（保証）までを一貫して辿れる構造を保ちながら、各 View 単体でも読みやすい粒度に抑えられる。

本書の 3View は、①意思決定の根拠（リスク起点）、②実装への落とし込み（配置と実現）、③運用上の実効性（保証ループ）を、それぞれ読みやすい粒度で示し、全体としてトレーサビリティを確保することを目的とする。

3.1 View A：意図と要件（なぜ／何を）

View A は、対象領域の「意味づけ」を行う視点である。環境変化や事業上の動機（Driver）、現状評価（Assessment：リスク・課題・機会）を起点に、設計思想（Principle）と満たすべき要求（Requirement）を整理し、達成状態（Outcome）や目標（Goal）へ接続する。

なお本書でいう「規範が果たすべき機能（Function）」は、規範が担う役割を言語化し、議論の焦点を揃えるための補助概念である。ArchiMate 上では、Function を独立した要素として追加せず、主として Requirement（および Constraint）と Outcome の記述として表現する。

ここでの成果は「議論の共通土台」であり、以降の統制設計（View B）や運用保証（View C）の検討を、同じ語彙と粒度で進めるための基準点となる。

この View のポイントは、以下の 3 つである。

① リスク起点で要求の必要性・強度を説明できる。
② 要求を「手続・組織運用で担保するもの（ガバナンス要求）」と「設計・実装の前提として担保するもの（設計要求）」に二層化することで、本人関与の強弱や境界条件（外部連携、委託、提供等）の違いを扱いやすくする。
③ 結果（Outcome）を明示することで、Outcome 志向の議論（何が達成されればよいか、どの程度満たせばよいか）を可能にする。

View A は、部門横断の合意形成における「最初に決めるべきこと」を固定する。ここで要素名・定義・関係（Influence 等）を揃えることで、後続の View で詳細化しても論点がぶれにくくなり、「実装が先行して目的が曖昧になる」「監査可能性が後付けになる」といった典型的な失敗を抑制できる。

3.2 View B：統制と実装（どこで／何で）

View B は、要求を現実の業務・システムに「配置」する視点である。対象となる情報や機能がどの工程・プロセス・連携で扱われるか（例：取得、保管、加工、分析、共有、提供、利用、廃棄など）を流れとして捉え、その上で統制（ポリシー、アクセス制御、暗号化、監視、ログ、データ最小化等）をどこに適用するかを示す。View A が「何を求めるか」なら、View B は「どこで効かせるか」を扱う。

この View のポイントは、以下の 3 つである。

① 要求が「どこで効くのか」を工程上に置くことで、抜け漏れ（特定工程だけ無防備）を見つけやすい。
② 統制をサービスとしてまとめ、実装（アプリケーション／技術）に Realization させることで、要求→統制→実装のトレーサビリティを確保できる。
③ 統制（サービス）と実装（実現手段）を分離して示すことで、特定技術ありきの議論を避けつつ、環境・制約の変化に応じた実装の代替・更新を可能にする。

線の密度が高くなる場合は、工程群をライフサイクルとして Grouping して統制を集約し、重要な境界（例：外部提供）だけ個別に詳細化する。これにより、モデルを「正確にするほど読めなくなる」というジレンマを回避し、読みやすさと正確さを両立できる。

3.3 View C：保証と更新（続けて守れるか）

View C は、統制の「実効性」を時間軸で担保する視点である。設計時の評価（例：影響評価、リスク評価、レビュー）と、運用時の監視・検知、是正、外部説明、監査・当局対応を循環として表し、統制が継続的に維持・改善される構造を示す。単に統制を置くだけでは、運用で形骸化しやすいという前提に立つ。

この View のポイントは、以下の 3 つである。

① 導入時に正しく見える統制でも運用で崩れるリスクを前提に、継続的な検証と是正を「設計に内蔵」できる。
② 説明責任を「証跡が生まれる経路」として可視化できる（何が記録され、誰が判断し、いつ外部説明に結びつくか）。
③ インシデント対応や監査を事後対応ではなく、設計の一部として位置づけられる。

Outcome 志向の整理においては、結果を満たし続けるための運用構造が不可欠である。View C は、要求が運用上どのように検証され、誰が判断し、何が記録され、どのタイミングで外部説明や監査につながるのかを一枚の構造として提示し、LE4SDS が重視する「実効性」を支える。

さらに、本 View では、インシデント、監査指摘、委託先変更、用途変更、外部ガイドライン更新等を「更新トリガ」として位置づけ、要求・統制・実装・証跡設計のいずれを見直すか（更新対象）を明示することで、構造が継続的に更新される状態を担保する。更新対象には、統制の設定や証跡設計に加え、規程・契約・ガイドライン等の規範自体の改定（条項・手続・責任分界の見直し）も含まれる。

4. ケーススタディ：個人情報保護ユースケースへの適用例

本章では、3 章で示した 3 つの View（View A：意図と要件、View B：統制と実装、View C：保証と更新）を、個人情報保護ユースケースに適用し、LE4SDS の考え方がどのように「構造」として表現され、読み取れるかを具体的に示す。

本章の ArchiMate モデルは、方法論を適用した例示（サンプル）であり、個人情報保護の全体を網羅することや、特定の解釈・運用の正確性を保証するものではない。目的は、論点を構造として整理し、検討・更新可能な形で共有するための読み取り方を示すことである。

4.1 ユースケースの概要（A 社・B 社）

本ケーススタディでは、A 社が保有する個人データを、B 社が統計処理・分析・AI 開発等の目的で取り扱う状況を想定する。ここで重要なのは、同じ「外部で処理する」状況であっても、取り扱いの前提（目的・裁量・技術的可能性）が異なると、本人不利益のリスクと求められる統制が大きく変わる点である。

本ケースでは、少なくとも次の二つのパターンを区別する必要がある。

- **委託型**：B 社は A 社の指示の範囲でのみ処理し、目的の追加や名寄せ（突合）等の裁量を持たない。
- **独自裁量が生じうる型**：B 社側で名寄せ（突合）や、将来の別目的での再利用が技術的・運用的に可能となり得る（例：B 社環境にデータが蓄積され、他データと結合できる／再識別やプロファイリングに繋がる余地がある等）。

LE4SDS の観点では、これらを形式的な区分（委託か提供か）だけで片付けるのではなく、「本人の権利利益に対する影響が何で、どの程度か」を起点に、達成すべき結果（Outcome）と統制の組み合わせを設計することが求められる。以降では、3 章で述べた 3 つの View（A/B/C）を用いて、この整理を「構造」として可視化する（図 1〜図 3）[1]。

[1] 本章で説明している各図の詳細を手元で確認したい場合は本書付属の ArchiMate ファイルを参照のこと

4.2 View A：リスク評価を起点に、要求（ガバナンス／設計）を整理する

図 1（View A）は、「なぜ／何を」を扱う View であり、Driver（動機・背景）と Assessment（評価：リスク）から出発して、Principle（原則）と Requirement（要求）を導出し、Outcome/Goal へ接続する構造を示している。

ここでのポイントは、(a)リスク起点で説明できること、(b)原則と要求を二層に分けることである。

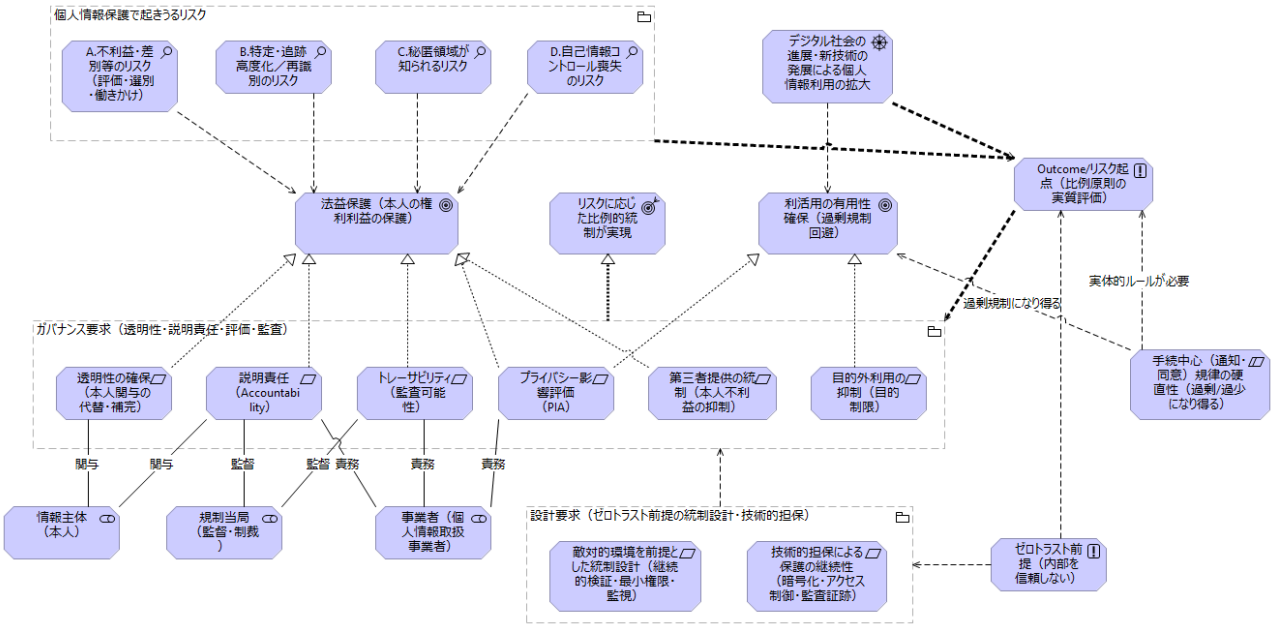

図 1　View A：意図と要件（動機・リスク・原則・要求）

まず、図 1 の Assessment（リスク群）として、本人不利益を複数の観点で整理する。例えば、名寄せ（突合）やプロファイリングによって本人の私生活が推知される、望まない働きかけが生じる、自己情報コントロールが失われる、といったリスクが代表例となる。委託型では、A 社の指示の範囲に収まるためリスクが相対的に抑制されるが、独自裁量が生じうる型では、B 社側での結合可能性や再利用可能性によって、同じデータでも影響の大きさ・広がりが増幅し得る。ここで重要なのは、「リスクの有無」ではなく、「リスクが高まる条件（境界・裁量・結合可能性）」を可視化することである。

次に、図 1 の Principle として、比例原則の実質評価（リスクに応じて統制強度を合理的に決め、説明可能にする）を規範の原則として置く。この Principle は、図 1 にまとめられているガバナンス要求（例えば、透明性、説明責任、PIA（影響評価）、監査可能性、目的制限、第三者提供統制、記録・証

跡など）を方向付ける。委託型であっても、境界を跨ぐ以上、透明性や説明責任が不要になるわけではなく、むしろ「なぜこの形でよいのか」を説明できる状態が求められる。

同時に、図 1 ではもう一つの Principle として、「ゼロトラスト（敵対的環境を前提とする設計）」を、設計前提の原則として位置づける。ここでいうゼロトラストは特定製品や構成を指すのではなく、「善意や境界防御に依存しない」ことを意味する。図 1 では、この Principle を、設計要求（例：敵対的環境を前提とした統制設計、技術的担保による保護の継続性）へ接続することで、ガバナンス要求が「実装可能かつ監査可能な形で成立する」ことを狙っている。

つまり図 1 は、

「リスク（Assessment）→ 原則（Principle）→ 要求（Requirement）→ Outcome/Goal」

という因果を示すだけでなく、原則を「規範の原則（比例原則）」と「設計前提（ゼロトラスト）」に分け、要求を「ガバナンス要求」と「設計要求」に分けることで、本人関与（同意等）だけに依存しない担保の必要性を説明可能にしている。

4.3 View B：統制をライフサイクルに配置し、統制と実装を分離する

図 2（View B）は、「どこで／何で」を扱う View であり、図 1 で定義した要求（ガバナンス要求・設計要求）を、実際の取り扱い工程と統制・実装へ落とし込む。

ここでのポイントは、(a)工程に置くことで抜け漏れを減らすこと、(b)要求→統制→実装のトレーサビリティを維持すること、(c)統制（サービス）と実装（アプリケーション／技術）を分離して示し、特定技術ありきの議論を避けつつ代替・更新可能性を確保することである。

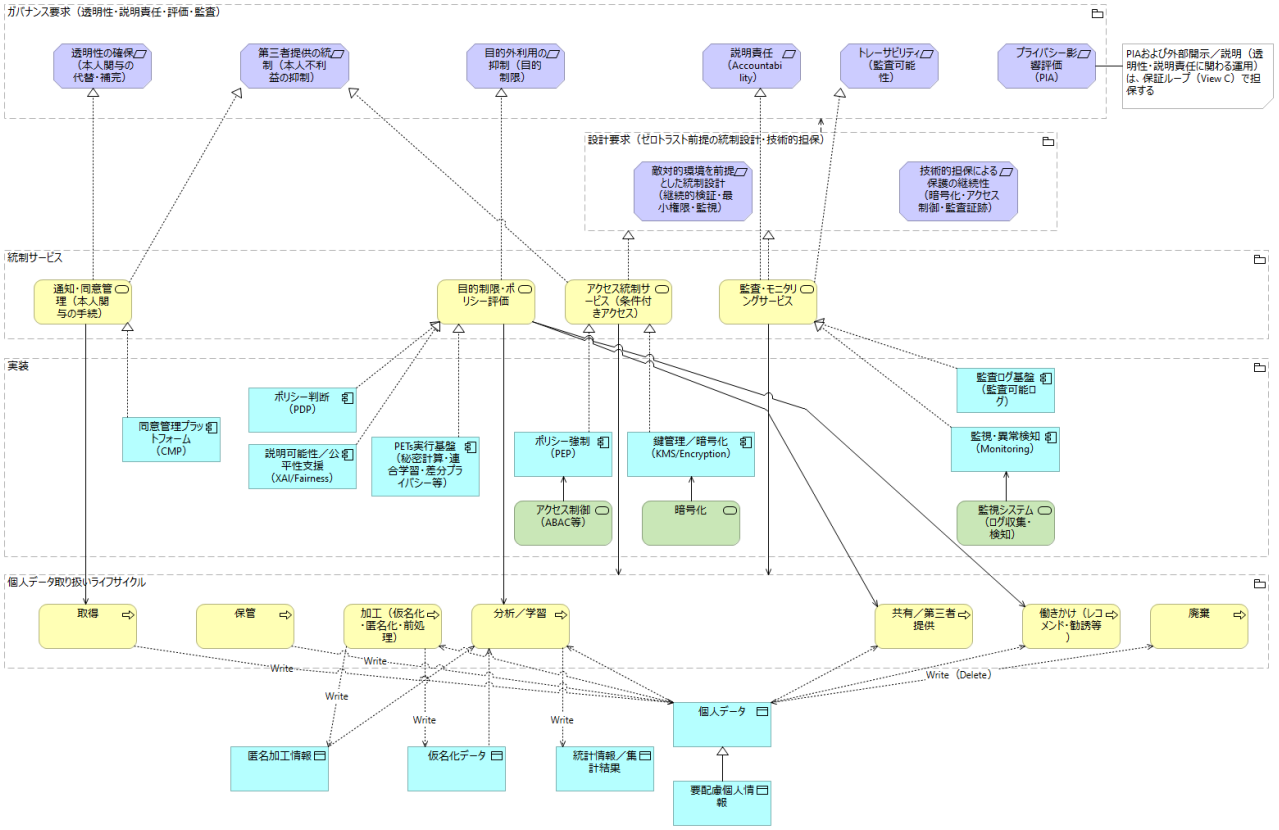

図 2 View B：統制と実装（ライフサイクル×統制×実装）

まず、図 2 では「個人データ取扱いライフサイクル」が Grouping されている（取得、保管、加工・分析、共有／第三者提供、利用（働きかけ）、廃棄等の工程を束ねたもの）。この Grouping を置く意義は、アクセス統制、監視・監査、ポリシー評価、暗号化等の統制が、特定工程だけでなく横断的に適用されることを、線の爆発を抑えつつ示す点にある。図 2 では、横断統制をライフサイクル Grouping へ集約し、重要な工程（差分が生じやすい工程）だけを個別に強調する構成が採られている。

次に、図 2 の統制サービス（Business Service 等）に着目すると、例えばアクセス統制（認可・最小権限・条件付きアクセス）、監視・監査（ログ・モニタリング・証跡）、ポリシー評価（目的制限・利用条件の判定）といった統制が、図 1 の要求に対応して配置されていることが分かる。委託型であっても、アクセス統制や監査証跡が欠ければ説明責任が成立しない。一方、独自裁量が生じうる型では、結合可能性や二次利用可能性が増すため、特に「保管（蓄積）」や「分析／学習」、「共有／第三者提供」の工程で、統制強度や統制の種類を厚くする必要が生じる（例：ポリシー評価の厳格化、鍵管理の分離、監査ログの改ざん耐性、監視の常時化など）。

このようにユースケースの型（委託型／独自裁量が生じうる型等）によって統制の追加・強化点が変わる場合、図 2 は共通骨格（ベースライン）として保持し、差分のみを抽出した追加の View（差分 View）として表現することが有効である。これにより、全体図の可読性を損なわずに、型ごとの検討事項（追加要求・強化点）を明確化できる。

さらに、図 2 では統制サービスが実装（Application/Technology）へ Realization される構造を持つ。これにより、要求が抽象論のまま浮遊せず、「どの統制が」「どの実装で」満たされるのかが辿れる。例えば、図 1 で監査可能性を要求するなら、図 2 では監査ログがどこで生成され、どの技術要素で保全されるかまで議論できる。こうした接続により、設計判断が「言っているだけ」にならず、実装・運用の整合へ落ちる。

要するに図 2 は、図 1 の要求を工程に配置し、統制（サービス）と実装（手段）を切り分けて実装へ落とすことで、「どこで何をすれば Outcome を支えられるか」を可視化する View となっている。

4.4 View C：同意以外で実効性を担保する保証と更新（保証ループ）

図 3（View C）は、「続けて守れるか」を扱う View であり、統制の実効性を時間軸（設計→運用→改善）で担保する。

ここでのポイントは、(a)導入時の正しさと運用時の継続性を分けて扱うこと、(b)説明責任を「証跡が生まれる経路」として可視化すること、(c)是正と監査を事後対応ではなく設計に内蔵することである。

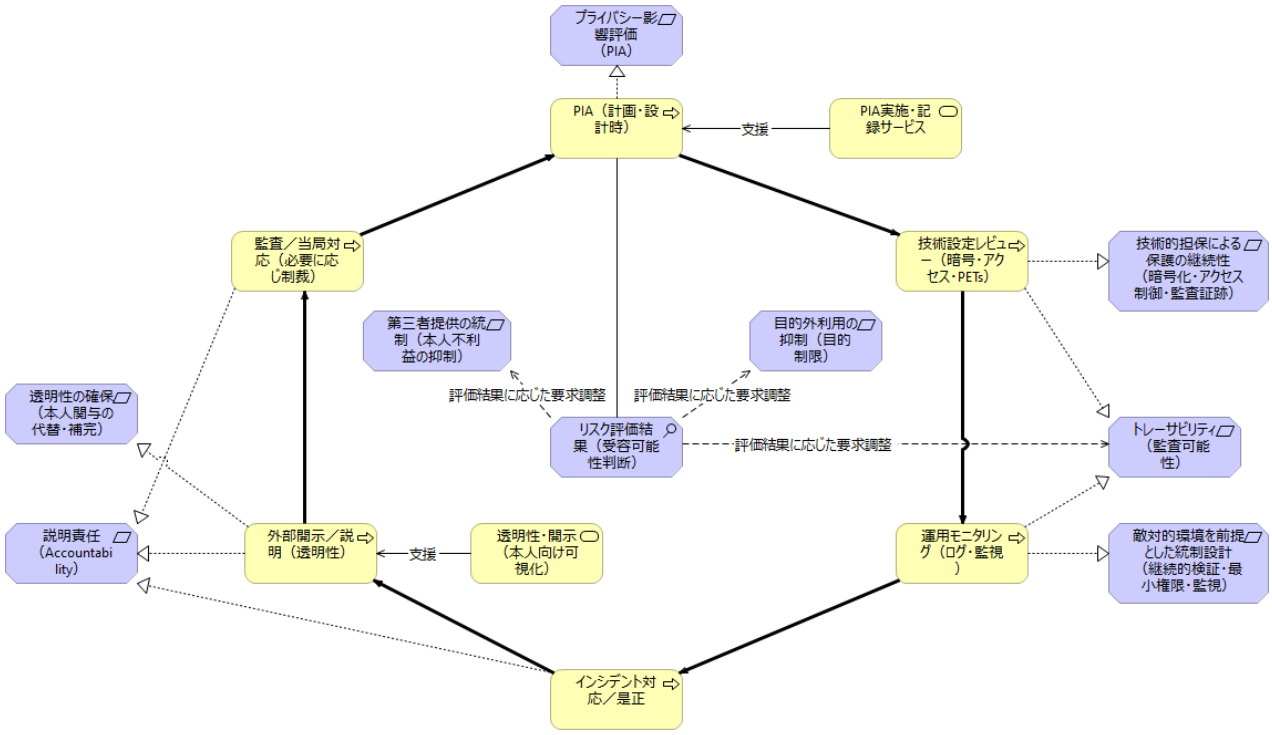

図 3　View C：保証と更新（評価・監視・是正・開示・監査）

図 3 では、PIA（影響評価）や技術設定レビュー（設計時の妥当性確認）が起点となり、運用モニタリング（ログ・監視）で状況を継続的に観測し、逸脱や異常が見つかれば是正（対策）へ繋ぎ、必要に応じて開示・説明（透明性）へ接続し、監査（第三者視点の検証）で保証する、という循環が表現されている。これは、同意や契約条項の整備だけでは担保しきれない領域（例えば運用での権限肥大化、例外対応の常態化、委託先の再委託、ログ運用の形骸化等）に対して、「維持される」ための構造を示すものである。

特に、図 1 で設計要求として掲げた「技術的担保による保護の継続性」は、図 3 の保証ループによって初めて実効性を持つ。暗号化やアクセス制御が「設定されている」だけでは不十分であり、設定が維持され、例外が監視され、逸脱が是正され、その記録が監査や外部説明に耐える形で残ることが必要となる。図 3 は、この「導入→維持→是正→説明→検証」のつながりを、運用プロセスとして示している。

要するに図 3 は、図 2 の統制が運用で崩れないようにするための「保証の循環」を定義し、図 1 の Outcome/Goal を満たし続けるための構造を提示する View となっている。

5. これから作る局面における有用性（考察）

本章では、前章までのケーススタディを踏まえ、構造の可視化が「既に決まったものを説明する」だけでなく、「これから作る／更新する」局面でどのように有効に働くかを考察する。以下の議論は、個人情報保護に限らず、規範要求（法令を含む）を設計・実装・運用に接続する一般の場面に適用可能な観点として整理する。

5.1 設計空間を整理し、意思決定を前倒しする

新規サービスやデータ連携を設計する段階では、目的・利用条件、取り扱うデータの性質と結合可能性、外部連携の範囲と責任分担、運用時の監視・監査要件など、論点が同時多発的に現れる。動機・リスク・原則・要求を先に構造化しておくと、何を議論すべきかが可視化され、設計の検討順序（例えば、まずリスク評価と Outcome、次に要求の二層化、最後に統制配置と保証）を揃えられる。（例：個人情報保護では、同意取得の可否、委託と提供の境界、名寄せの可能性、外部委託先の統制水準などが論点となる。）

ここでいう「前倒し」とは、設計の早い段階で「論点を固定する」ことである。例えば、リスク評価が曖昧なまま統制案だけを議論すると、後から「前提が違った」「想定していたユースケースが違った」となり、要求や実装の手戻りが起きやすい。View A で Assessment→Principle→Requirement→Outcome の関係を先に置くと、議論の焦点が「手段の好き嫌い」ではなく、「どのリスクに対し、どの Outcome を、どの強度で担保するか」に揃う。結果として、View B/ C の議論も、前提のズレが少ない状態で進められる。

5.2 トレードオフを「結果（Outcome）」として扱える

利活用の有用性と法益保護はしばしば対立するが、Outcome として「どの程度の結果を達成したいか」を置くことで、統制強度の選択を「手続の有無」ではなく「結果の担保」として議論できる。このとき、比例原則（リスク起点の実質評価）を Principle として明示し、ガバナンス要求と設計要求を二層化しておくと、設計選択の根拠を維持しやすい。

Outcome を置く意義は、トレードオフを「可視化された選択」として残せる点にある。利活用の価値を高めるほどリスクが増える場面では、統制を一律に強化するだけではコストや利便性が破綻し得る。比例原則の実質評価を Principle として明示し、Outcome（達成したい状態）に照らして要求（ガバナンス要求／設計要求）を組み立てることで、統制強度が「禁止／許可」ではなく「結果の担保」として説明可能になる。これは、関係者や監査に対して「なぜその落とし所なのか」を再説明する際にも有効である。

5.3 保証ループを最初から設計に含める

技術統制は導入時点では正しく見えても、運用で形骸化しやすい。保証ループ（評価・監視・是正・開示・監査）をモデルとして組み込み、どこで証跡が生まれ、誰が判断し、いつ是正されるかを事前に定めることは、同意に依存しない実効的保護の前提となる（例：委託先変更や用途変更をトリガに要求・統制を見直す、など）。

保証ループは、統制を「導入したら終わり」にしないための設計である。実務では、例外対応、権限の肥大化、委託先の変更、ログ運用の形骸化などにより、統制は時間とともに崩れる。View C で、評価（設計の妥当性）→監視（運用の観測）→是正（逸脱の復元）→開示（説明）→監査（検証）を循環として置くと、技術的担保が「継続的に成立している」ことを説明しやすくなる。特定の手続や一時点のチェックに依存せず、運用の実態として保護が成立していることを示すために、この循環を前提に設計することが重要である。

5.4 適用上の留意点

最後に、適用上の留意点を三つ述べる。いずれも、モデルの品質を「正しさ」だけでなく「読みやすさ」「更新され続けること」で確保するための観点である。

⑴ View を分け、集約を使う

1 枚に詰め込みすぎない。線密度が上がる場合は、Lifecycle Grouping 等で横断統制を集約し、リスクや要件が相対的に高まる工程、または差分が生じる箇所だけ詳細化する。

⑵ 要素名・粒度を揃える

同義語の乱立を避け、「原則」「要求」「統制」「実装」を混同しない。特に、要求（What）と統制（How）を分けて記述する。

⑶ 更新責任を設計する

モデルは更新されて初めて価値が出る。版管理と更新責任（オーナー）、更新トリガ（ユースケース追加、委託先変更、事故発生等）を明確にする。

6. まとめ

本書は、LE4SDS の考え方（リスク起点・Outcome 志向・ガバナンス×技術による実効性確保）を、「構造（因果・トレーサビリティ・運用循環）」として可視化・共有するための方法論を整理し、その具体例として ArchiMate を用いて説明した。可視化の狙いは、結論（統制や手続）の妥当性だけでなく、「なぜそれが必要か」「なぜその強度か」「運用で維持できるか」を説明可能にする点にある。

提案した枠組みは、View A（意図と要件）で意思決定の根拠を固定し、View B（統制と実装）で工程と統制・実装を接続し、View C（保証と更新）で継続的な実効性を担保する、という三つの視点から構造を保持するものである。これにより、部門横断の合意形成、説明責任、変更耐性の向上を同時に狙える。

個人情報保護ユースケースへの適用例では、原則の二層化（比例原則／ゼロトラスト）と要求の二層化（ガバナンス要求／設計要求）を核に、統制の配置と保証ループまで含む構造として表現できた。今後は他ユースケースへ横展開し、Outcome やリスク評価の粒度、監査観点・証跡設計との接続を洗練させることで、より実務的な活用（設計の前倒し、手戻り削減、継続的な改善）に繋げられると考えている。

以上