1-419
.NET Framework と Web サービスを考慮したシステム設計手法 (Part II)

コンポーネントの関係と接続技術の適用の基本

アジェンダ

- Part1とPart2の関係
- 開発者と設計者の視点、
  フラットランドとホラーキー
- フラットランドとしての .NETテクノロジー
- ホラーキーとしての .NETテクノロジー
- フラットランドに最適な開発工程モデル　RAD
- ホラーキーに最適な開発工程モデル
  ウォーターフォール
- RADとウォーターフォールの組み合わせ
- Call for Action

Part1とPart2の独立した関係

- Part1
  - ソフトウェア開発工程モデルの重要性
    - フラットランドとホラーキー
    - .NETテクノロジーをそれぞれの観点から説明
    - フラットランドに最適な開発工程としてのRAD
    - ホラーキーに最適な開発工程としての
      ウォーターフォール
- Part2
  - コンポーネントの関係
  - コンポーネント間接続の問題と接続技術

コンポーネントの関係

- ソフトウェアの単位すべてを
  ここではコンポーネントと呼ぶ
- コンポーネントは、複数のコンポーネントが組み合わされて、システム全体を構成する
- システム全体は、現代ではネットワーク全体と同義である
- この際に、コンポーネントの相互関係には、
  一定の原則が存在している
- 基本的な原則であるので、手法化はできないが、設計者にとって判断基準となる

20の原則　01

- リアリティ(実在/現実)を構成するものは、データでもプロセスでもなく、コンポーネントである
  - あらゆるシステムは、コンポーネント/オブジェクトが複合的に構成して成立している
    - コンポーネント化がされていないシステムであっても、1つの大きなコンポーネントとして成立している
    - コンピュータ化されていない部位であっても、インタフェースが特殊なコンポーネントして理解することが出来る
      - 人による認証等

20の原則　02

- コンポーネントには4つの基本的な性格がある
  - 自己保存
    - 自己独自の全体性や自立性を保存するための、自己の独自性(エイジェンシー)を保つ性格
  - 自己適応
    - 全体の中の部分として、他のコンポーネントに適応する性格
  - 自己超越(または自己変容)
    - 他のコンポーネント組み合わされて、新しい能力をもつコンポーネント群(コミュニオン)を実現する
  - 自己分解
    - 自己超越の逆の経路を辿り崩壊する

20の原則　02

20の原則　03

- **コンポーネントは発生する**
  - 自己超越により
    新しいコンポーネントが発生する
    - もとのコンポーネントからのみ完全に導き出せない性質や属性を持っている
    - 新しく発生したコンポーネントを、要素であるコンポーネントに還元することはできない
    - 高位に位置するコンポーネントのルールを、低位に位置するコンポーネントのルールで説明することは出来ない
    - 不確定的で非決定的な性格が内在している

20の原則　04

- **コンポーネントは**
  ホラーキー（階層的）に発生する
  - 高位のコンポーネントは、
    先行する低位のコンポーネントを包含する
  - コンポーネントは、水平、
    垂直の関係で自然な位置を占める
  - 極端な単一コンポーネント化、極端な平等的
    関係コンポーネント群を、不自然という

20の原則　05

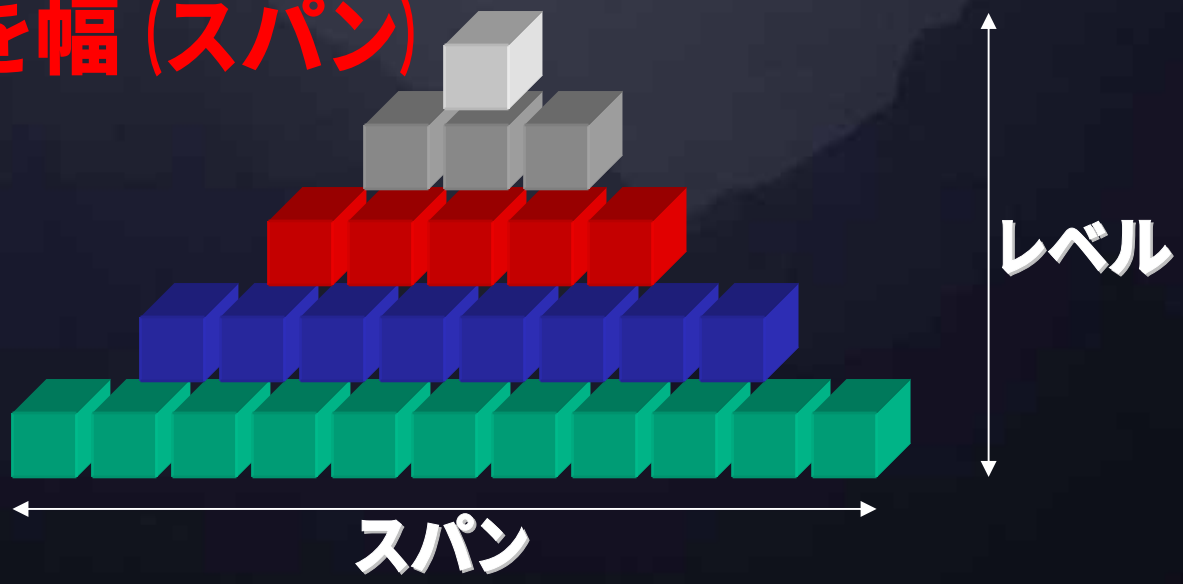
- 発生したコンポーネントは
  先行するコンポーネントを超越し、
  包含する
  - 低位のコンポーネントは
    すべて高位の中にある
    しかし、高位のコンポーネントは
    低位の中に無い

20の原則　06

- 低位のコンポーネントは
  高位のコンポーネントの可能性を定め
  高位のコンポーネントは
  低位のコンポーネントの確立性を定める
  - 高位のコンポーネントは低位のコンポーネントを超越していても、低位のコンポーネントのレベル、法則を犯すことは出来ない
    - 高位のコンポーネントの存在により、低位のコンポーネントの仕様は確定的になってく。

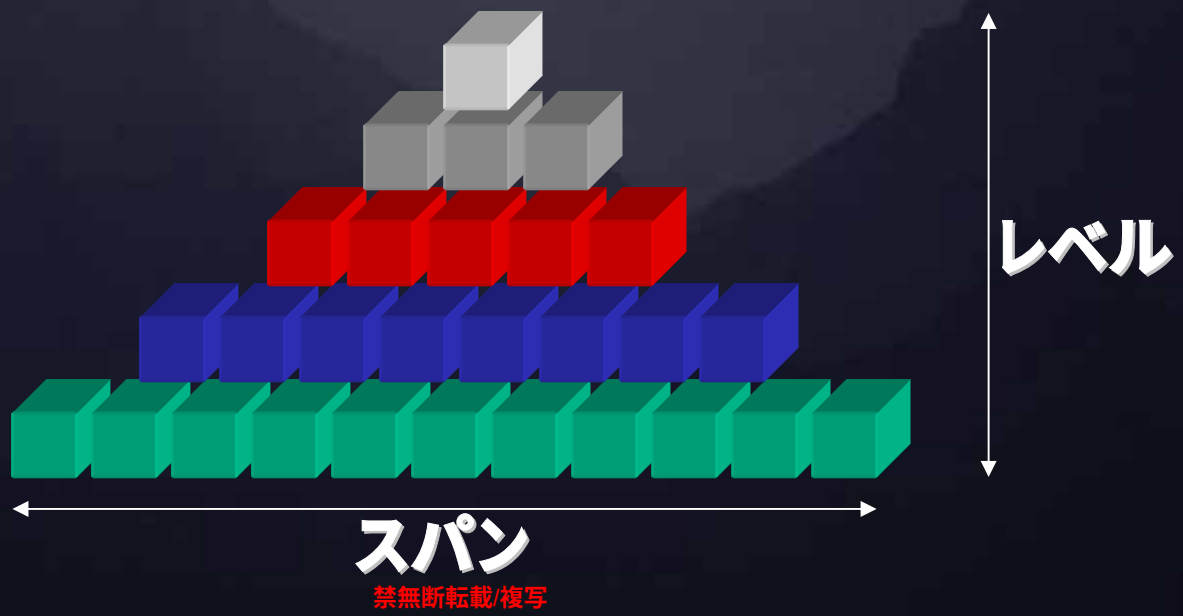
20の原則　07

- 階層が包含するレベルの数が、その階層が深いか、浅いかを決定する。
そして所与のレベルにおけるコンポーネントの数を幅（スパン）という



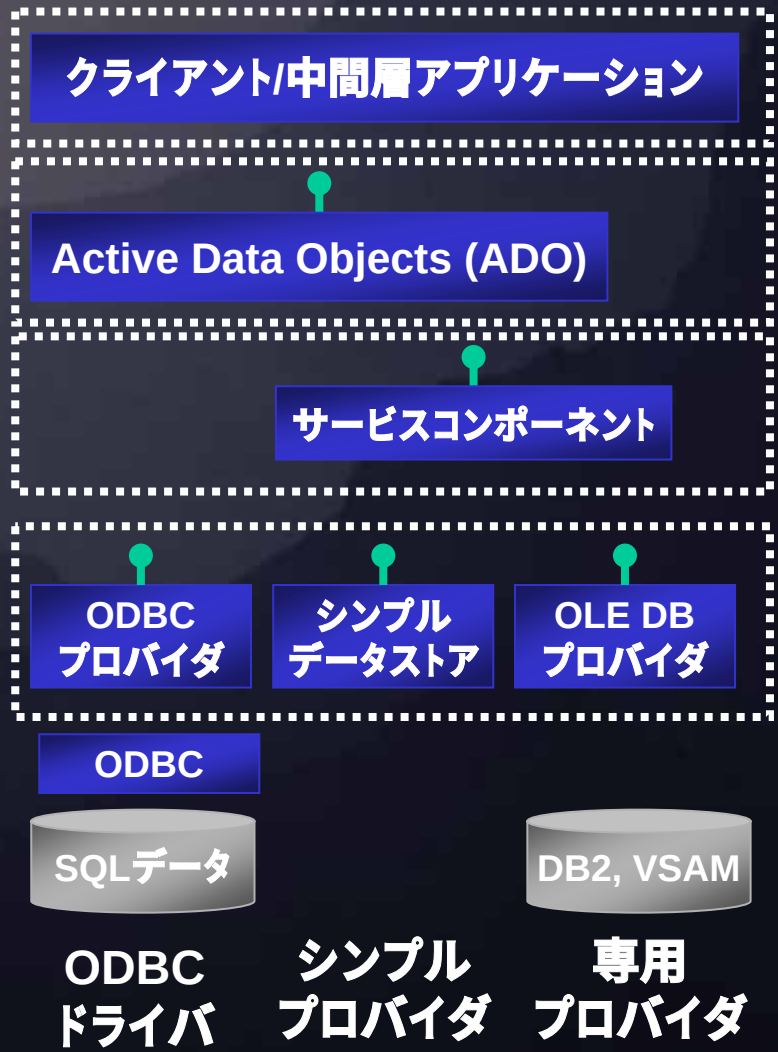
20の原則　08

- 連続的に発生する進化は、
  より深い深度と、
  より狭い幅(スパン)をもつ



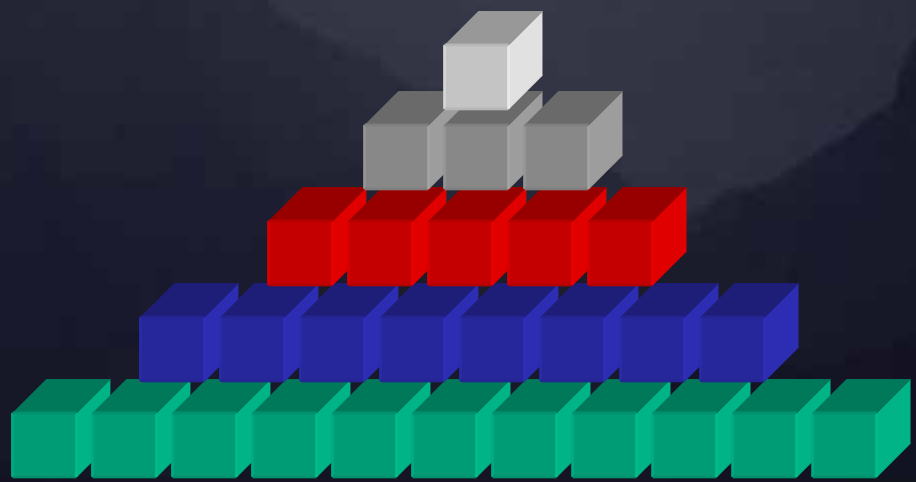
20の原則　09　(付加原則1)

- コンポーネントの深度が深くなれば(より、高いレベルになれば)、基本的な概念の深度も増大する(より高度になる)。



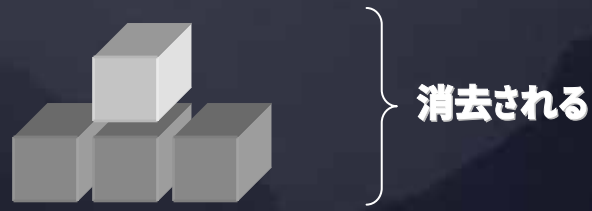
20の原則　10

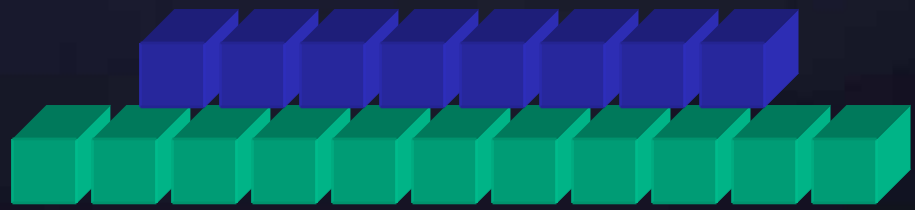
- どのコンポーネントのレベルを消去しても、それより上のコンポーネントはすべて消去され、それより下のコンポーネントは消去されない



20の原則　10

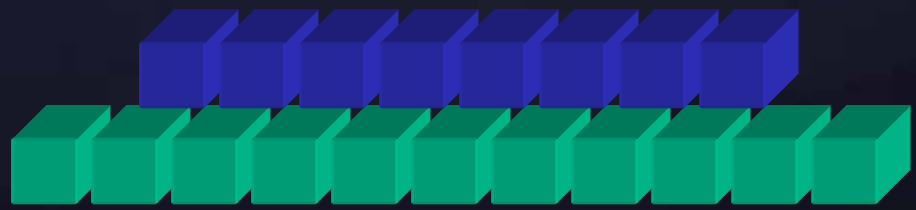
- どのコンポーネントのレベルを消去しても、それより上のコンポーネントはすべて消去され、それより下のコンポーネントは消去されない





20の原則　10

- どのコンポーネントのレベルを消去しても、それより上のコンポーネントはすべて消去され、それより下のコンポーネントは消去されない



20の原則　11

- **コンポーネントのホラーキー(階層)は共進化する**
  - コンポーネントが孤立して進化することは無い。
    孤立したコンポーネントが存在しないからである。進化は生態的(エコロジカル)である。

20の原則　12

- ミクロは、深度のあらゆるレベルでマクロと相関的な交換を行う
  - 「同レベルの相関的交換」
    コンポーネントが進化するにつれ、そのコンポーネントが含む階層のコンポーネントは、構造的な組織の同じレベルにあるほかのコンポーネントとの関係のネットワークの中に存在し、またその存在に依存している。
  - すべてのコンポーネントは先行するコンポーネントの複合体であり、それに明白な自己独自の発生的なパターンを付け加える。コンポーネントのそれぞれのレベルは、社会的 (マクロ)環境にある同じ深度のコンポーネントと相関的な交換を行うことでその存在を維持している

20の原則　13

- **進化は方向をもつ**
  - 進化が進む限り以下の方向を持っている
  - a. 複雑性の増大
  - b. 差異化/統合の増大
  - c. 組織化/構造化の増大
  - d. 相対的な自立性の増大
  - e. テロスの増大

20の原則　14

- 進化の方向　**複雑性の増大**
  - 新しいレベルの複雑性は、異なった角度からは新しい単純性を伴う
  - 高位のシステムの発生は･･･システム機能の単純化でもある

20の原則　15

- 進化の方向　差異化/統合の増大
  - 進化は差異化と統合が一緒になって働く
  - したがって、常に差異化/統合と説明する
  - 両者は対極にあるようだが、一方が増大すれば他方が減少するだけである

20の原則　16

- 進化の方向　**組織化/構造化の増大**
  - 上位のシステムの進化が、自己を規定するシステムのレベルの更なる複雑化に結びつく
  - 単純なタイプのシステムから、複雑なタイプのシステムへ、低レベルの組織から、高レベル組織へと進んでいく

20の原則　17

- 進化の方向　**相対的な自立性の増大**
  - 相対的な自立性とは、自己保存(エイジェンシー)の別な言い方
  - 環境的な揺らぎの中でのコンポーネントの自己保存能力が増大すること
  - コンポーネントの深度が深い(レベルが高い)と、相対的な自立性は増大する
    - 頑健であることを示しているわけではない

20の原則　18

- 進化の方向　テロスの増大
  - テロスとは、深層構造の終わりに導く小さな意味でのオメガ・ポイント
    言い換えると短期的目標で、正しい最後の目標に至る道標に相当するもの

20の原則　19(付加原則2)

- **すべてのコンポーネントは関係している全体に対してIOU(借用書/前提条件)を発行する**
  - すべてのコンポーネントは、完結性と一貫性が両立しない。コンポーネントが部分として完結しようとすれば、他と整合性の無い不安定なものになっていき、またその逆もありうる。
  - すべてのコンポーネントは、関係している全体に対して不完全であるか、不整合である。
    そのためにIOUを切る。

20の原則　19(付加原則2)

- 具体例
  - データベースは単体で完結する設計である
    - アプリケーションが使用する際に、使用しにくい
    - リアルタイム環境に対して独特の配慮が必要
    - システムリソースを最小にする方法が無い
    - オーバーヘッドが大きい
  - データオブジェクトを使用すると、特定目的に最適化することしか出来ない
    - 使用しやすいが、あらゆる用途には耐えられない
    - あらゆる用途に耐えるようにすると、オーバースペックになる

20の原則　20(付加原則3)

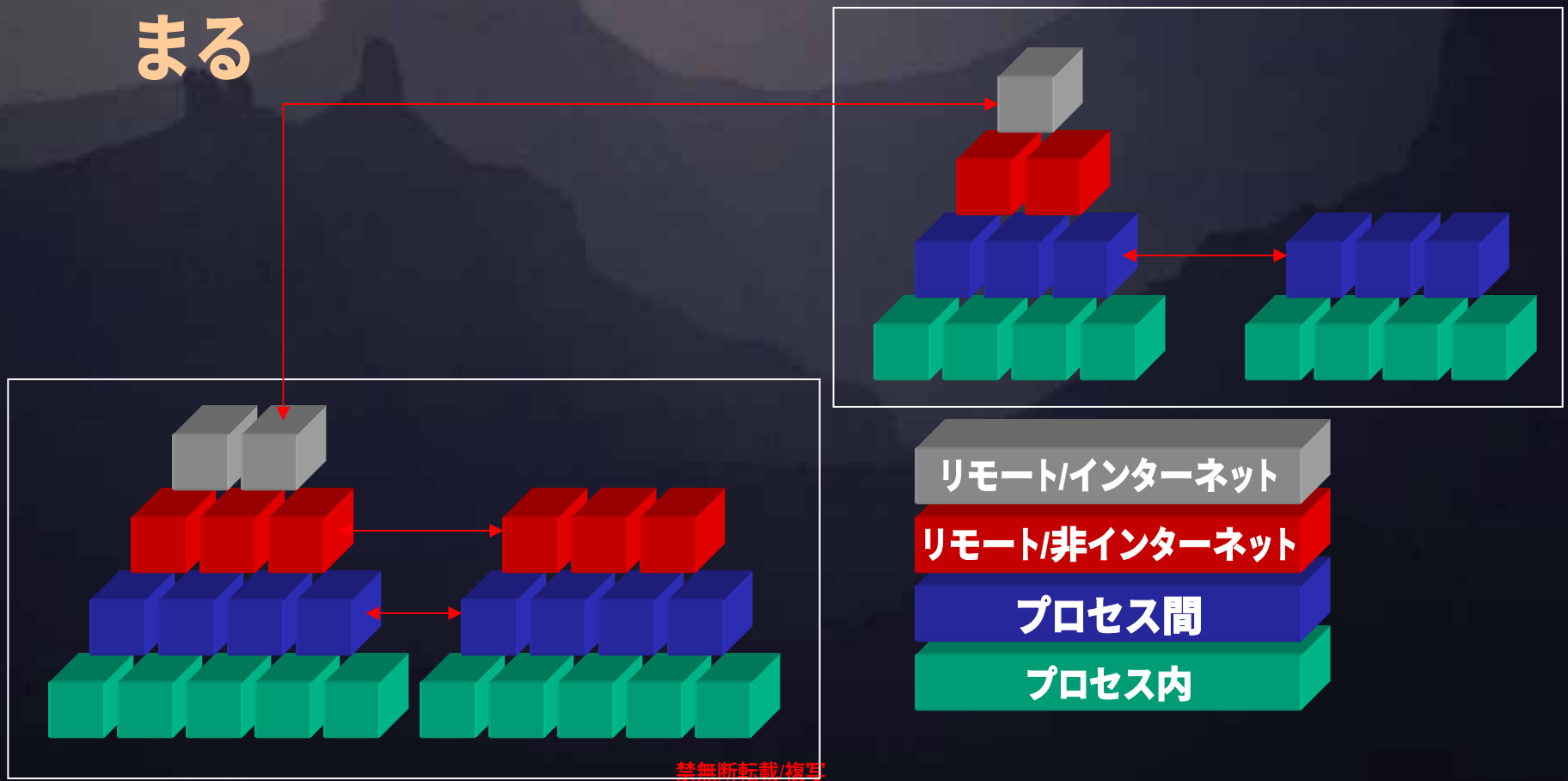
- すべてのIOUは
  コンポーネントの(恒常的な)存在そのものによって、清算される
  - IOUの相関関係を考えると、無制限な義務の連鎖という問題に紛れ込んでしまう
  - この問題は、コンポーネントの存在そのものによってのみ清算することが出来る。

コンポーネント間接続の問題

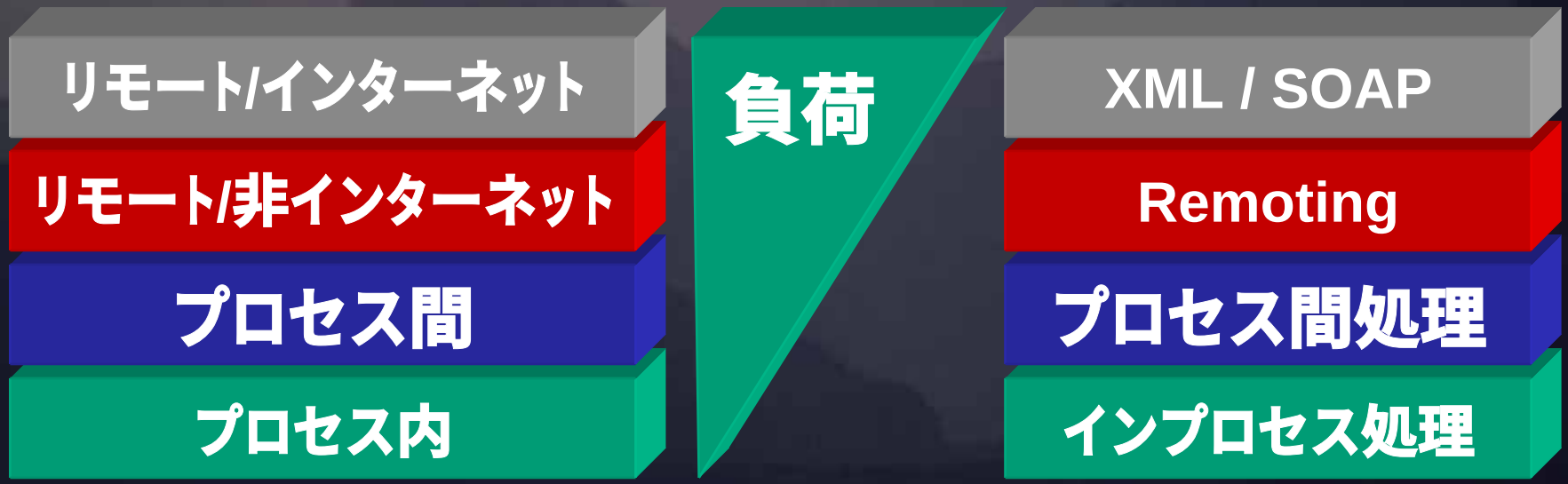
- コンポーネントの接続には負荷が伴う
  - 物理的負荷
  - インタフェースに伴う負荷
- 負荷に対応した解決策を適切に採用
  - 採用に失敗があると、現実的なシステムは開発できない

物理的負荷の性格

- 物理的ホラーキーにより負荷の性格が決まる



物理的負荷と採用するテクノロジ



物理的ホラーキーの他の側面

- ネットワークスピード
  - 高速なほど負荷が低い
  - 同一システムがもっとも負荷が低い
- メモリリソース
  - 仮想記憶を使用しないだけのメモリ容量がもっとも負荷が低い
- CPUリソース
  - CPUが速いほど負荷が低い

インタフェースに伴う負荷

- ベリー・アーリー・バインデング
  - .NETアプリケーション
  - COM標準/カスタム・マーシャリング
- アーリー・バインデング
  - .NETアプリケーション
  - ユニーバーサル・マーシャリング
- レイト・バインデング
  - XML / SOAP
  - ネイティブ・オートメーション・プログラム

参考資料

- ホロン階層/ホラーキー（holarchy）
  - HierarchyとHolonを組み合わせた単語
    Arthur Koestlerの作り出した用語で、今では広く使用されている
- ホロン（Holon）
  - 人間が理解する単位すべてを指しており、全体の中の部分も、全体も、それぞれすべてをホロン(Holon)とよぶ
  - ソフトウェアコンポーネントもすべてホロンである
  - 動的システム理論で、原則が導出された
  - 本資料はKen Wilberの理論に準拠

資料:ホロン階層 20の原則 ①

- リアリティを構成するものは、
  モノでもプロセスでもなく、ホロンである
- ホロンは4つの基本的な力を持っている。
  自己保存、自己適応、自己超越、自己分解の力である
- ホロンは発生する
- ホロンはホロン階層的に発生する
- 発生したホロンは先行するホロンを超越し、
  包含する

資料:ホロン階層 20の原則 ②

- 低位のホロンは高位のホロンの可能性を定め、高位のホロンは低位のホロンの確立性を定める。
- 階層が包含するレベルの数が、その階層が深いか、浅いかを決定する。
  そして所与のレベルにおけるホロンの数を幅（スパン）という
- 連続的に発生する進化は、より深い深度と、より狭い幅(スパン)をもつ

資料:ホロン階層 20の原則 ③

- ホロンの深度が深くなれば、意識の深度も増大する。
- どのホロンのレベルを消去しても、それより上のホロンはすべて消去され、それより下のホロンは消去されない
- ホロン階層は共進化する
- ミクロは、深度のあらゆるレベルでマクロと相関的な交換を行う

資料:ホロン階層 20の原則 ④

- 進化は方向をもつ
  - a. 複雑性の増大
  - b. 差異化/統合の増大
  - c. 組織化/構造化の増大
  - d. 相対的な自立性の増大
  - e. テロスの増大
- すべてのホロンはコスモスに対してIOU(借用書)を発行する
- すべてのIOUは「空性」において清算(=救済)される