SPLE における導入時の課題

～派生開発との違い、部分適用のやり方、構成管理など～

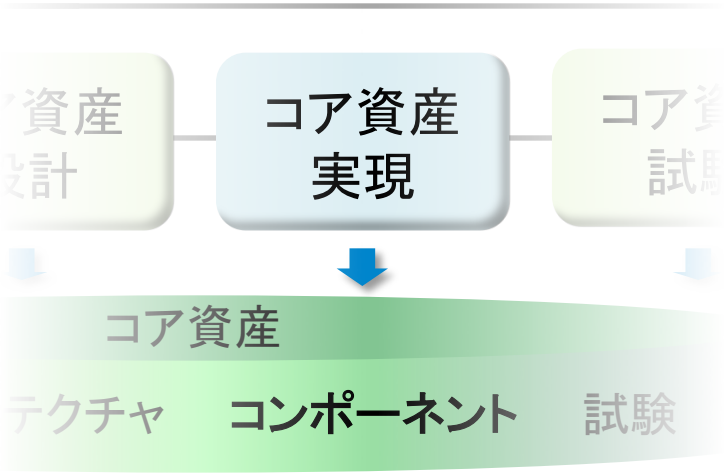
2013年03月06日

JASPIC プロダクトライン分科会

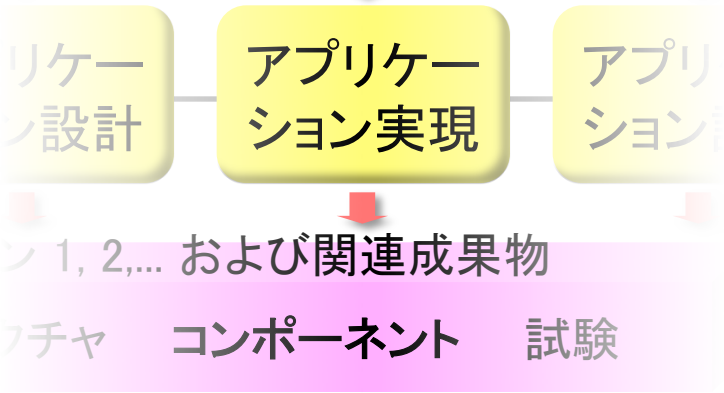
トレンド ― Partial SPLE

ドメイン開発

- 一部の資産を PL 化
- その後、範囲を広げる
  - 範囲：同種の成果物の他の部分／他の成果物の種類／他のサブプロセス／他のプロジェクト／…



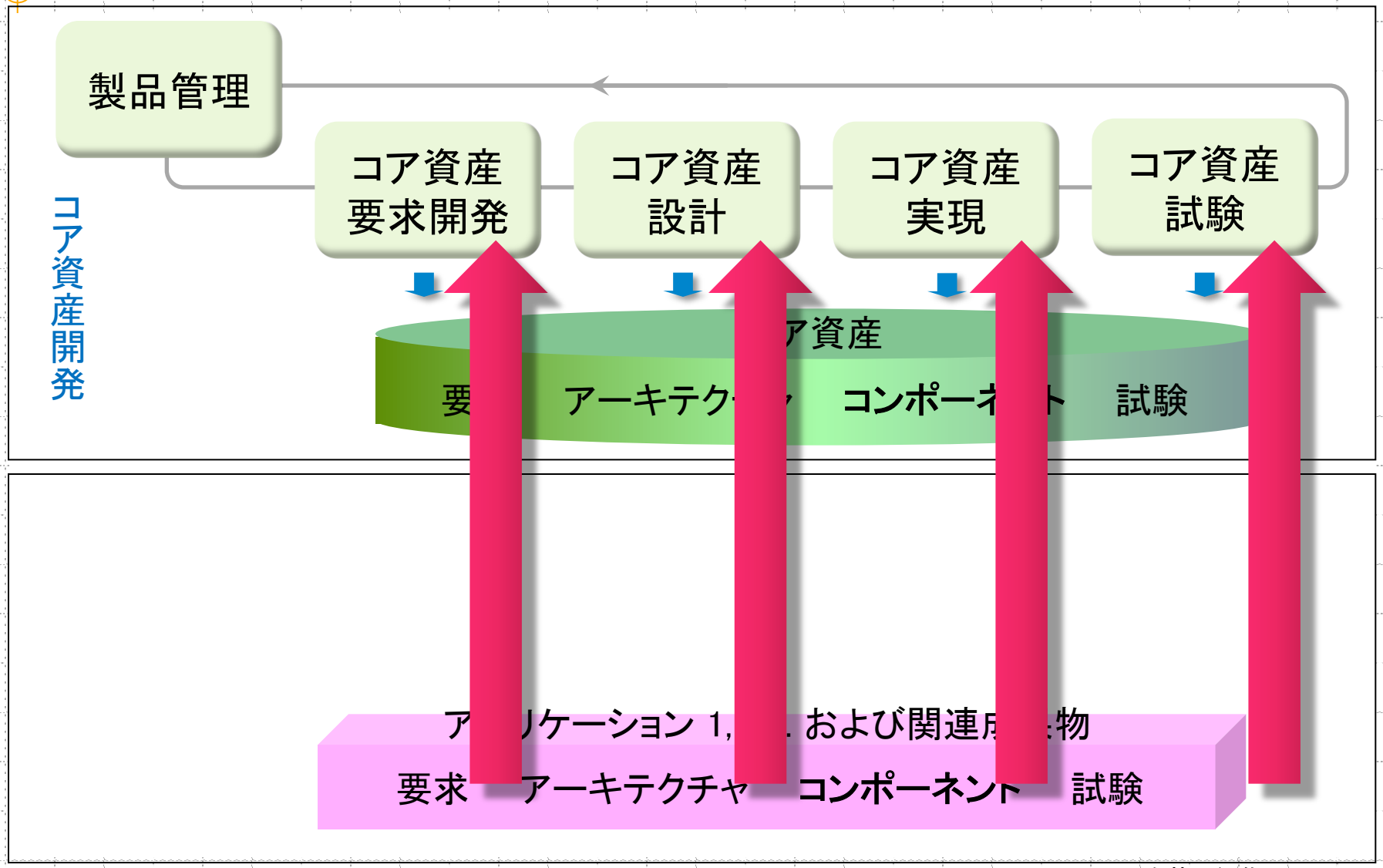
アプリケーション開発



現実的アプローチ ― 抽出式 SPLE

- 既存の開発成果物から共通化できる/すべきものを抽出しコア資産第1版とする
  - 必要に応じて可変性を持たせて
- 以後はそれを進化させていく
  - 可変性の追加
  - 新規コア資産の追加
  - コア資産の保守

現実的アプローチ ― 抽出式 SPLE

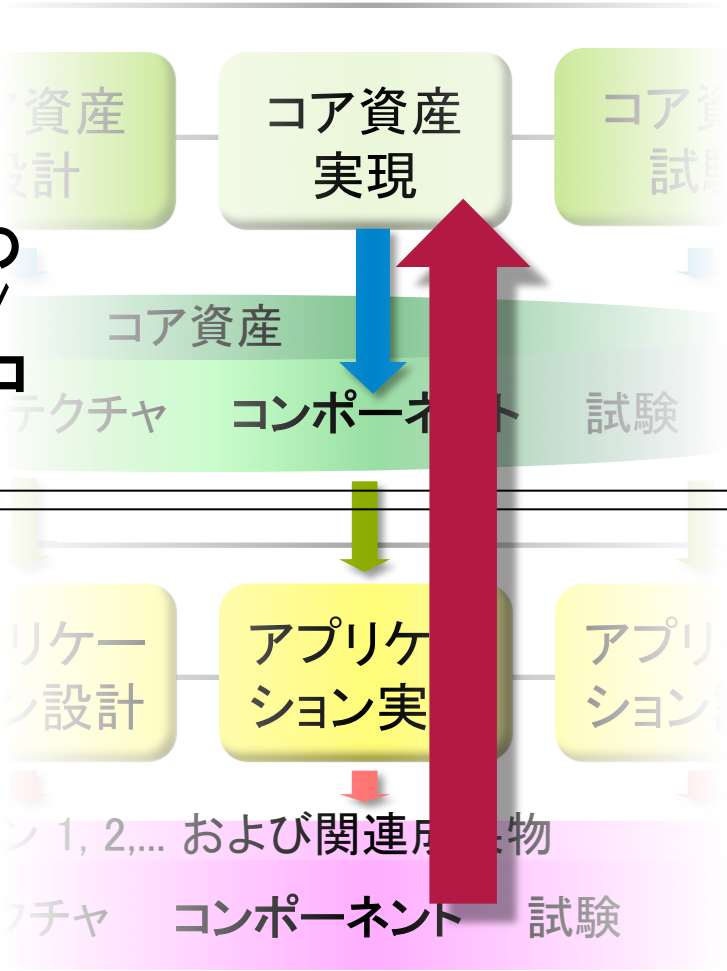


[Pohl05]を基に編集。

トレンディな現実的アプローチ ― Partial+抽出式 SPLE

コア資産開発

- 一部の資産を抽出式で PL 化
- その後、範囲を広げる
  - 範囲：同種の成果物の他の部分／他の成果物の種類／他のサブプロセス／他のプロジェクト／…



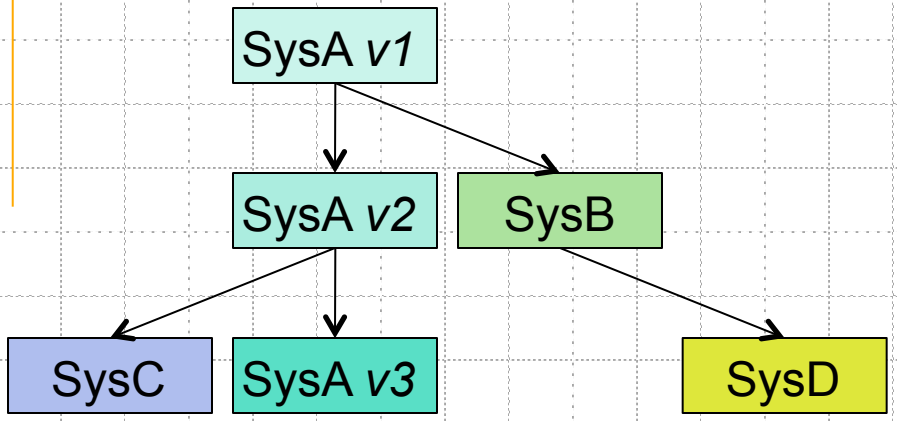
アプリケーション開発

詳細設計/ソースコードからコア資産にする例 →

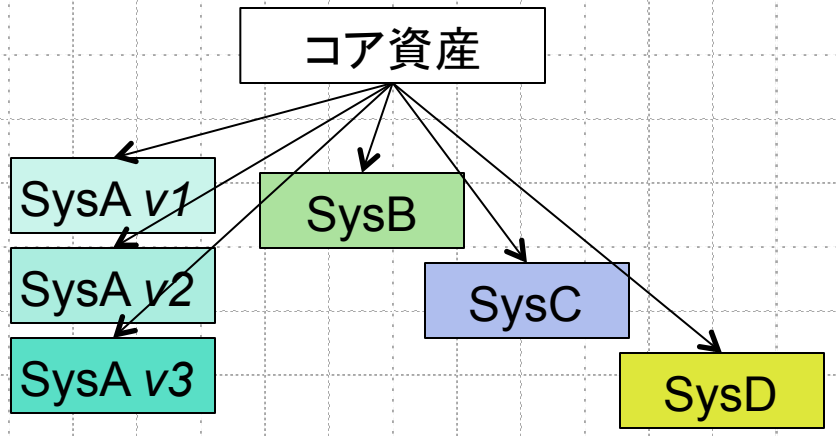
派生開発と SPLE

派生開発：差分開発、改造開発、改良保守、等とも呼ばれてきた

派生開発の典型例

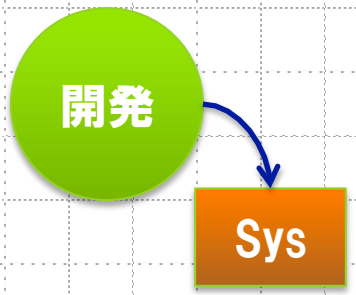


SPLEの基本形

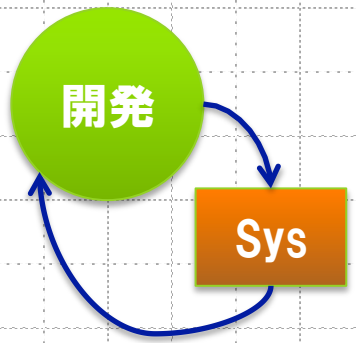


- 過去のシステムを基に再利用を検討する
  - 作ったものを再利用する
- 将来のシステムを基に再利用を検討する
  - 再利用するものを作る

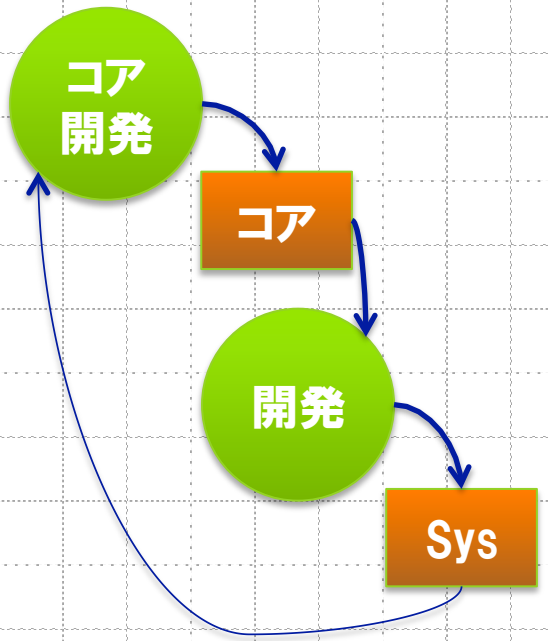
開発形態三種



単一開発

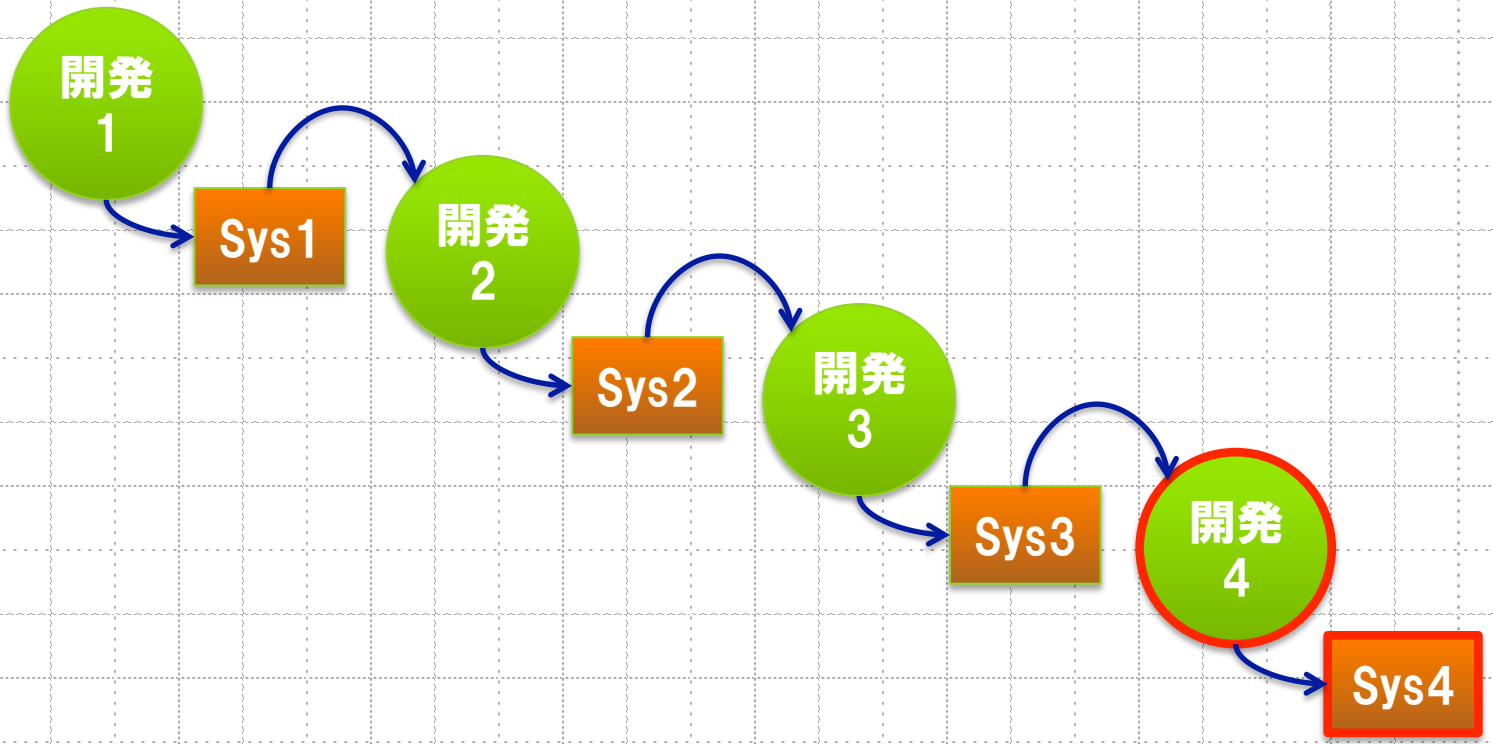


派生開発



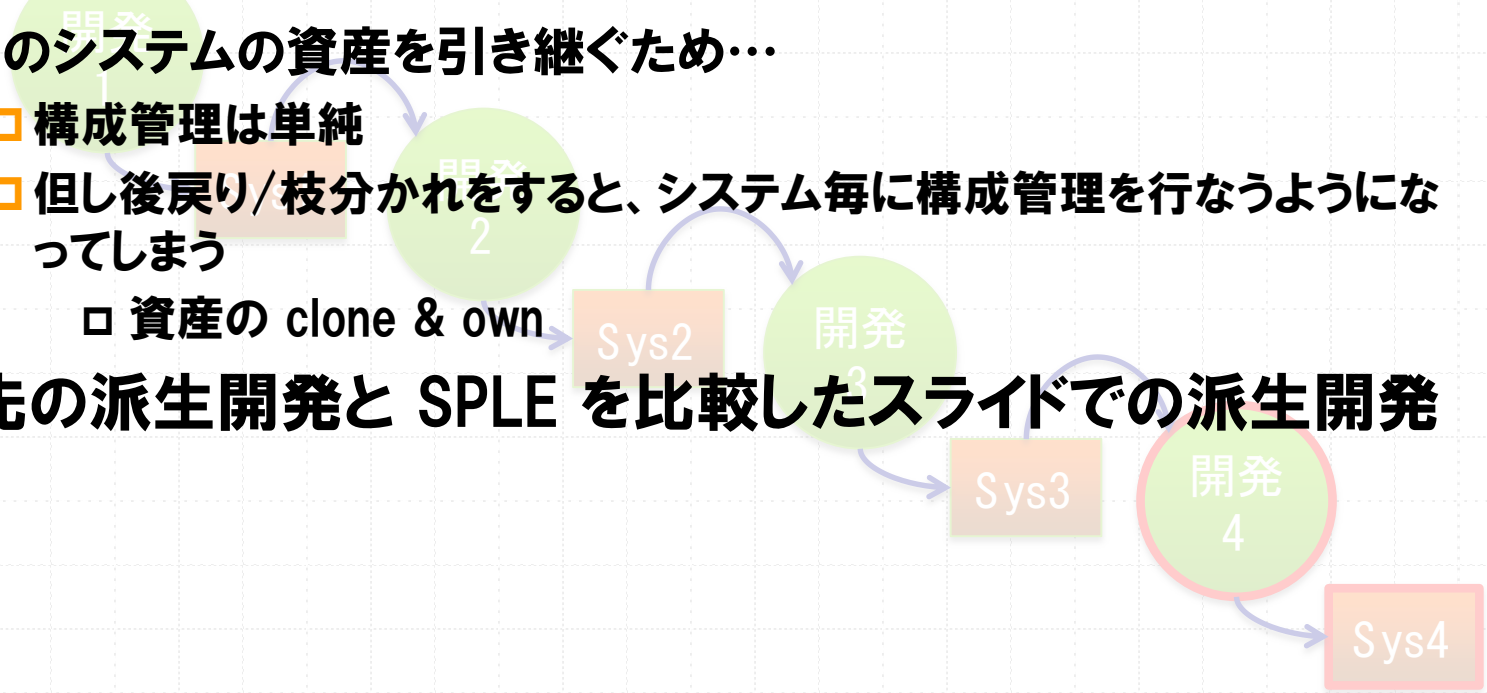
SPLE

派生開発の様子



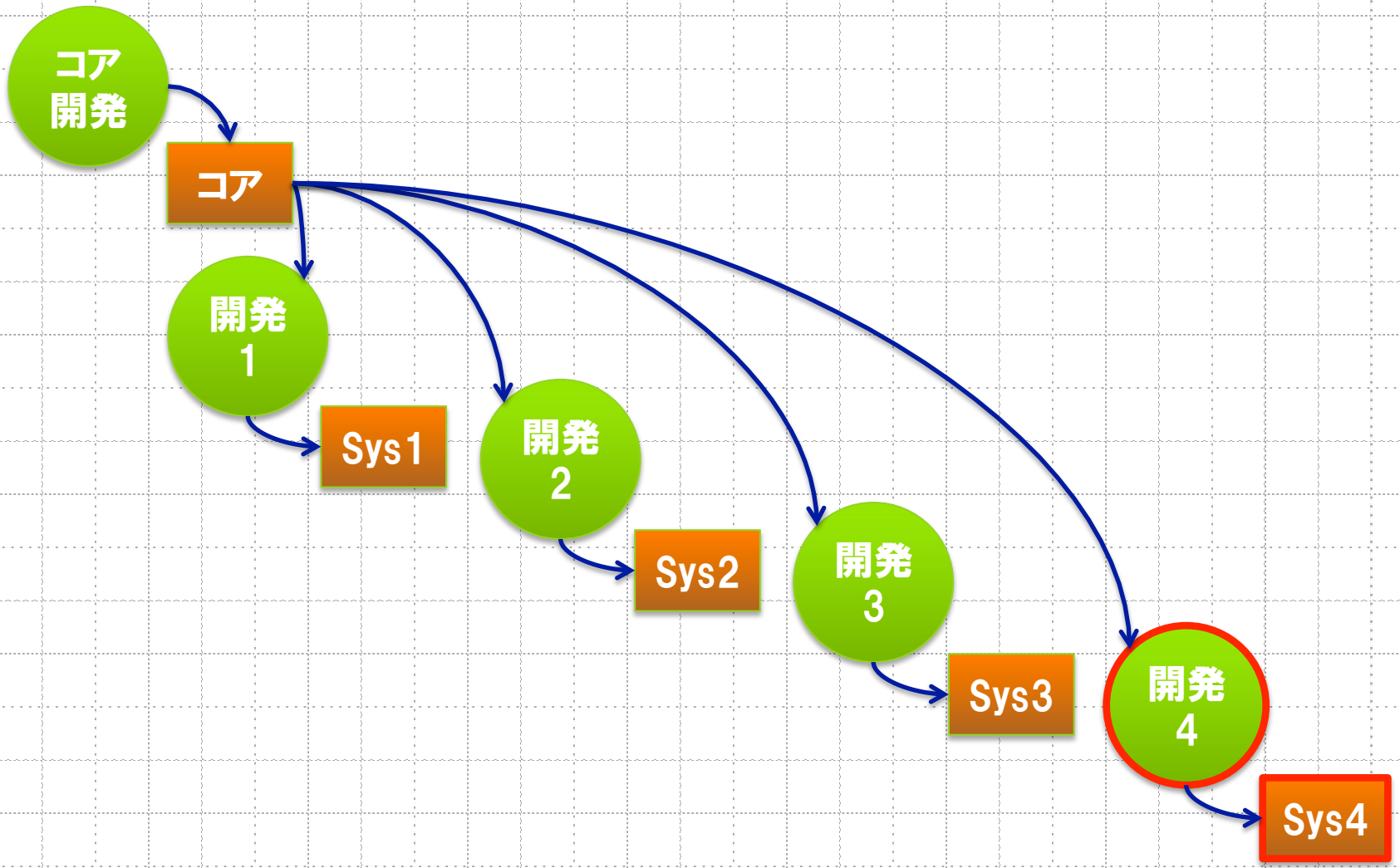
派生開発の特徴

- 準備のコスト・期間: ゼロ
  ↓
- アプリケーション開発コスト・期間は小さい（筈）
- 資産管理コスト:
  - 前のシステムの資産を引き継ぐため…
    - 構成管理は単純
    - 但し後戻り/枝分かれをすると、システム毎に構成管理を行なうようになってしまう
      - 資産の clone & own



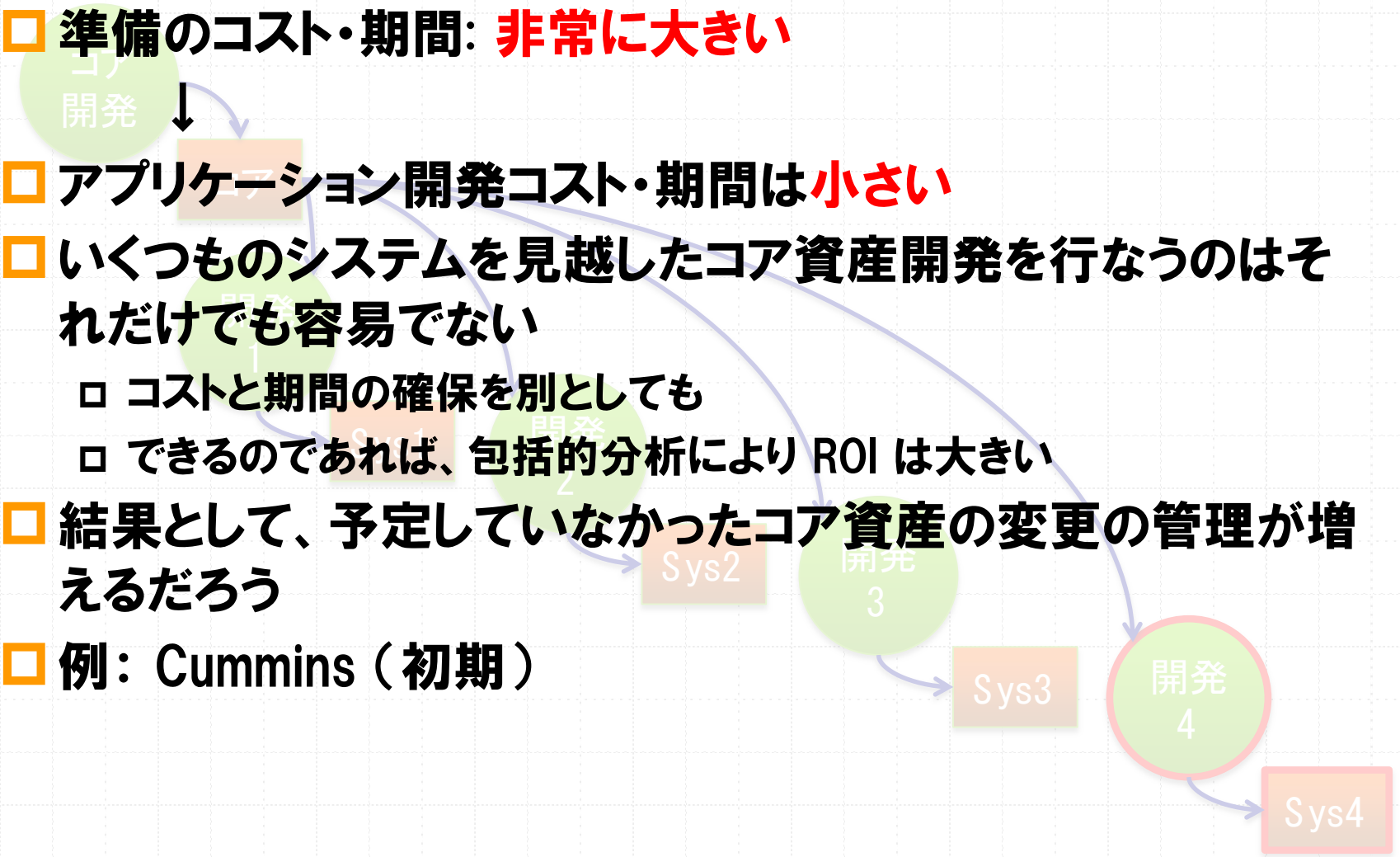
- 例: 先の派生開発と SPLE を比較したスライドでの派生開発

SPLE(事前準備式)の様子

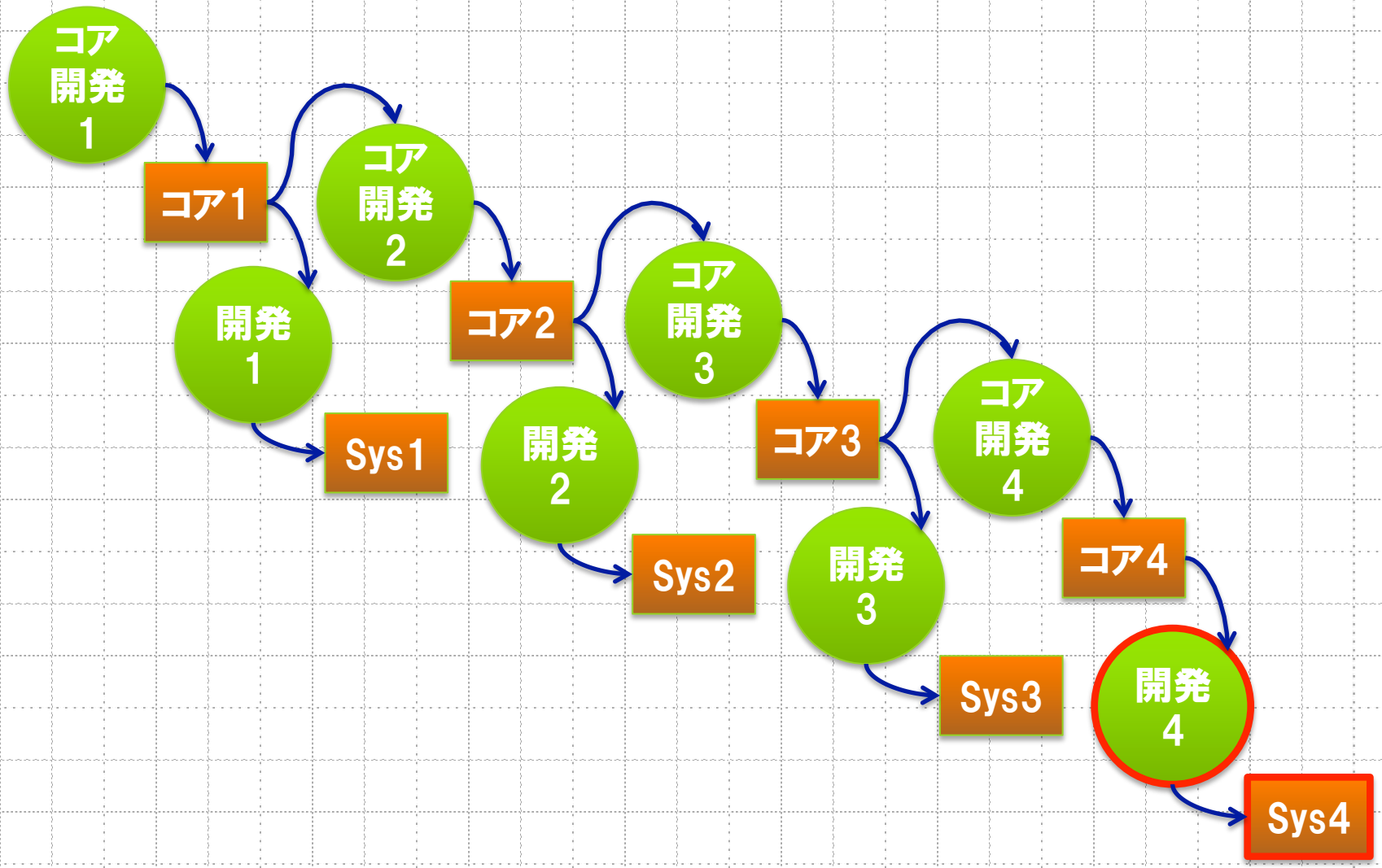


SPLE(事前準備式)の特徴

- 準備のコスト・期間: 非常に大きい
  ↓
- アプリケーション開発コスト・期間は小さい
- いくつものシステムを見越したコア資産開発を行なうのはそれだけでも容易でない
  - コストと期間の確保を別としても
  - できるのであれば、包括的分析により ROI は大きい
- 結果として、予定していなかったコア資産の変更の管理が増えるだろう
- 例: Cummins (初期)

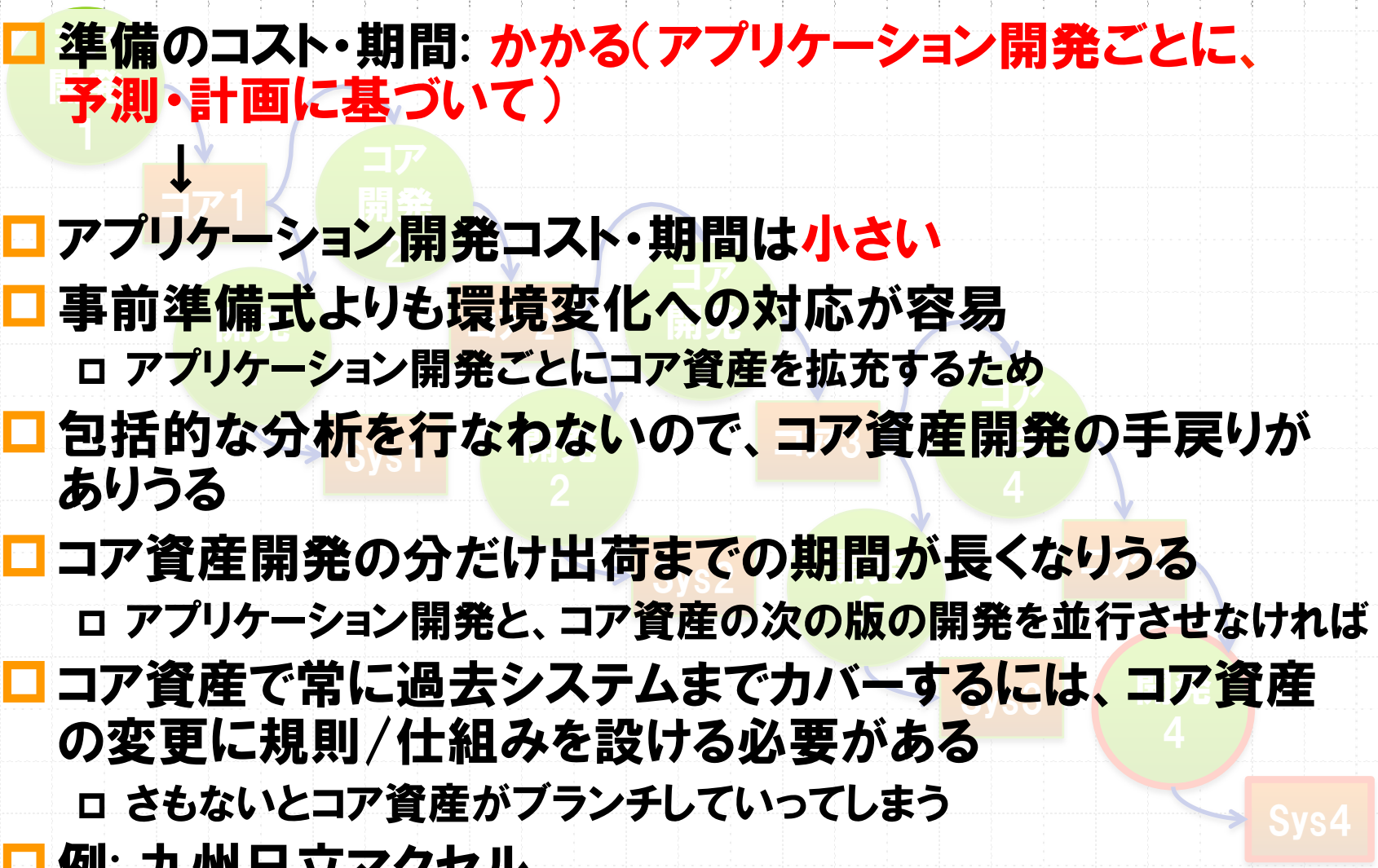


SPLE(事前都度対応式)の様子

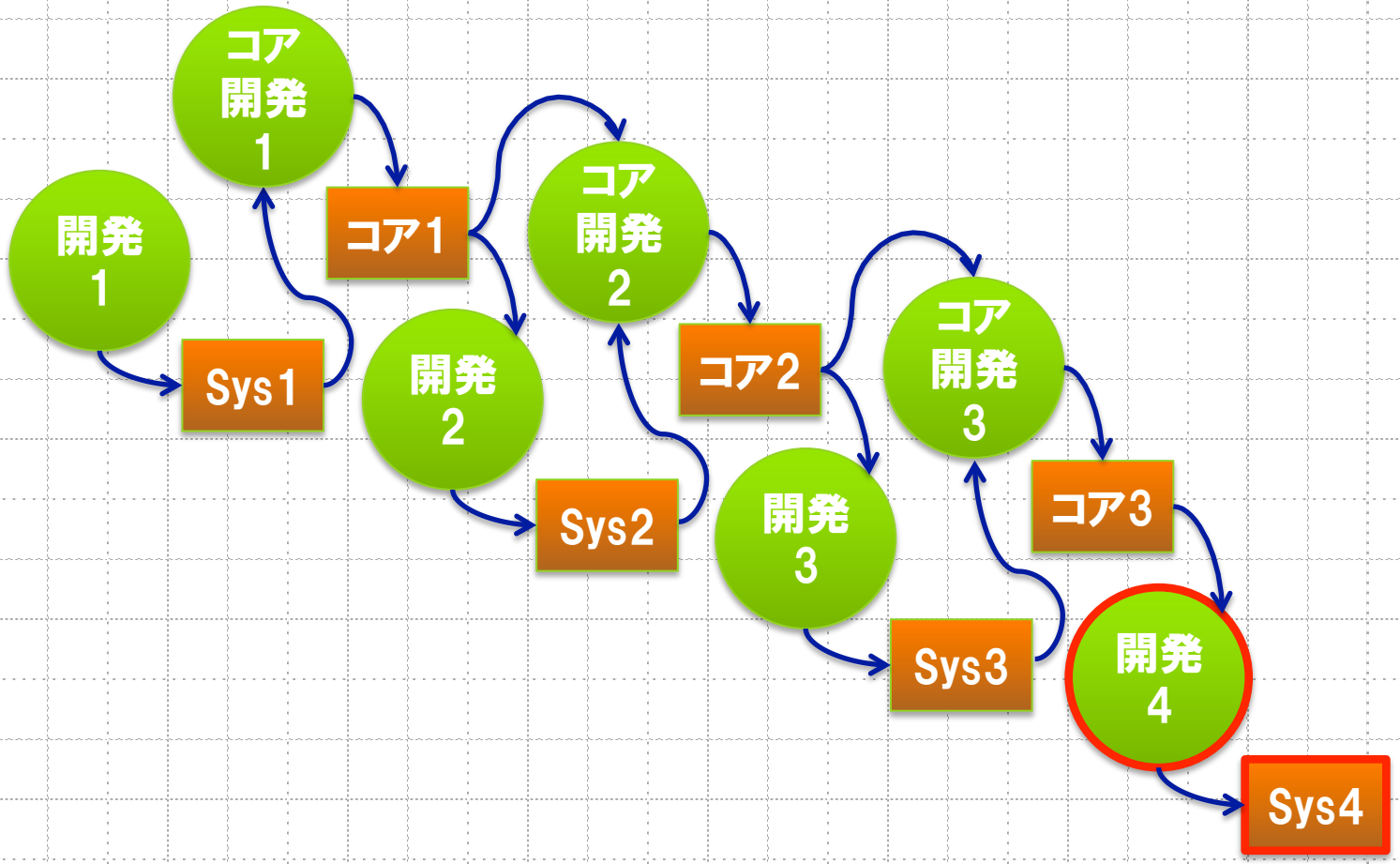


SPLE(事前都度対応式)の特徴

- 準備のコスト・期間: かかる(アプリケーション開発ごとに、予測・計画に基づいて)
  - ↓
- アプリケーション開発コスト・期間は小さい
- 事前準備式よりも環境変化への対応が容易
  - アプリケーション開発ごとにコア資産を拡充するため
- 包括的な分析を行なわないので、コア資産開発の手戻りがありうる
- コア資産開発の分だけ出荷までの期間が長くなりうる
  - アプリケーション開発と、コア資産の次の版の開発を並行させなければ
- コア資産で常に過去システムまでカバーするには、コア資産の変更に規則/仕組みを設ける必要がある
  - さもないとコア資産がブランチしていってしまう
- 例: 九州日立マクセル



SPLE(事後都度対応式)の様子

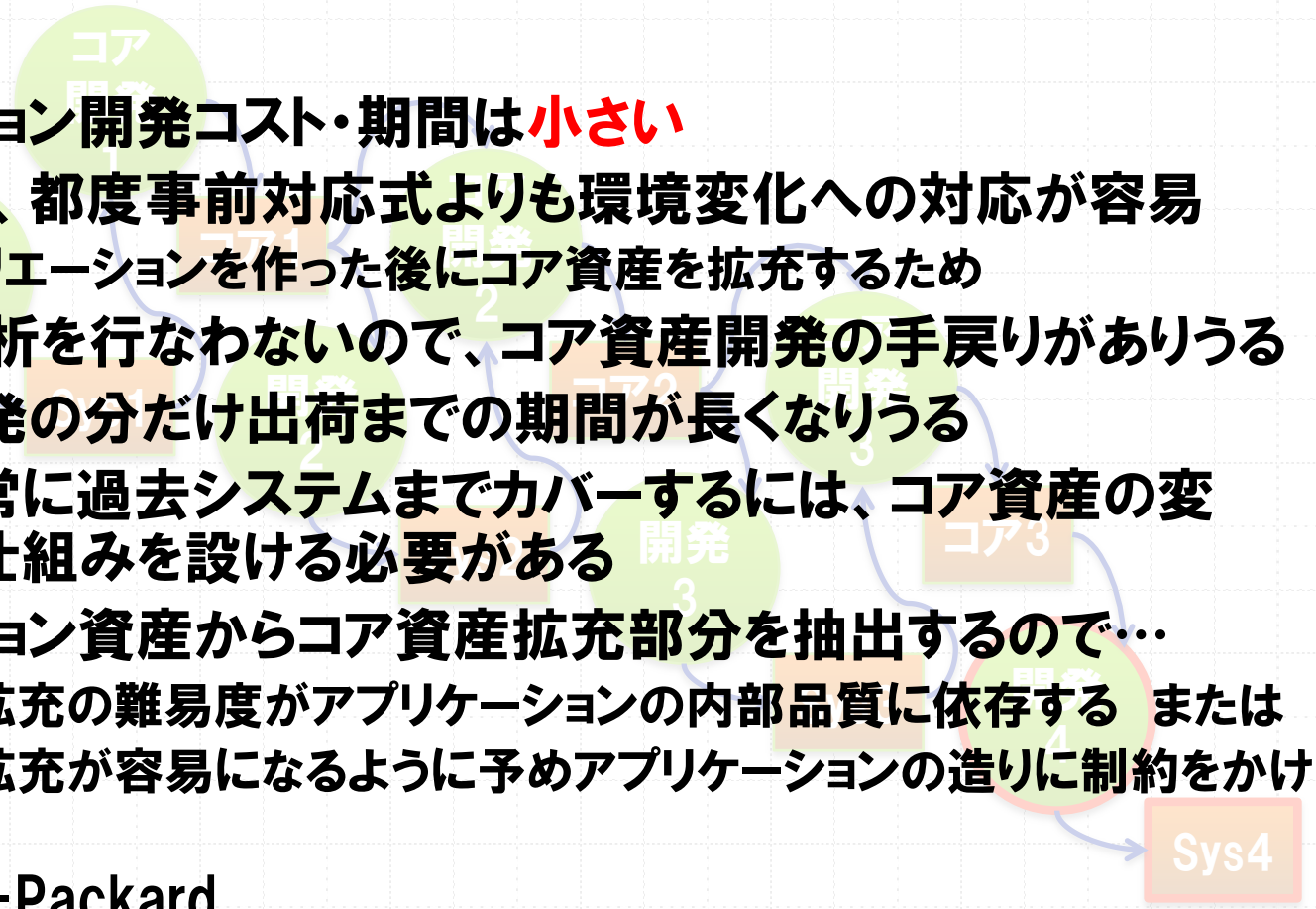


SPLE(事後都度対応式)の特徴

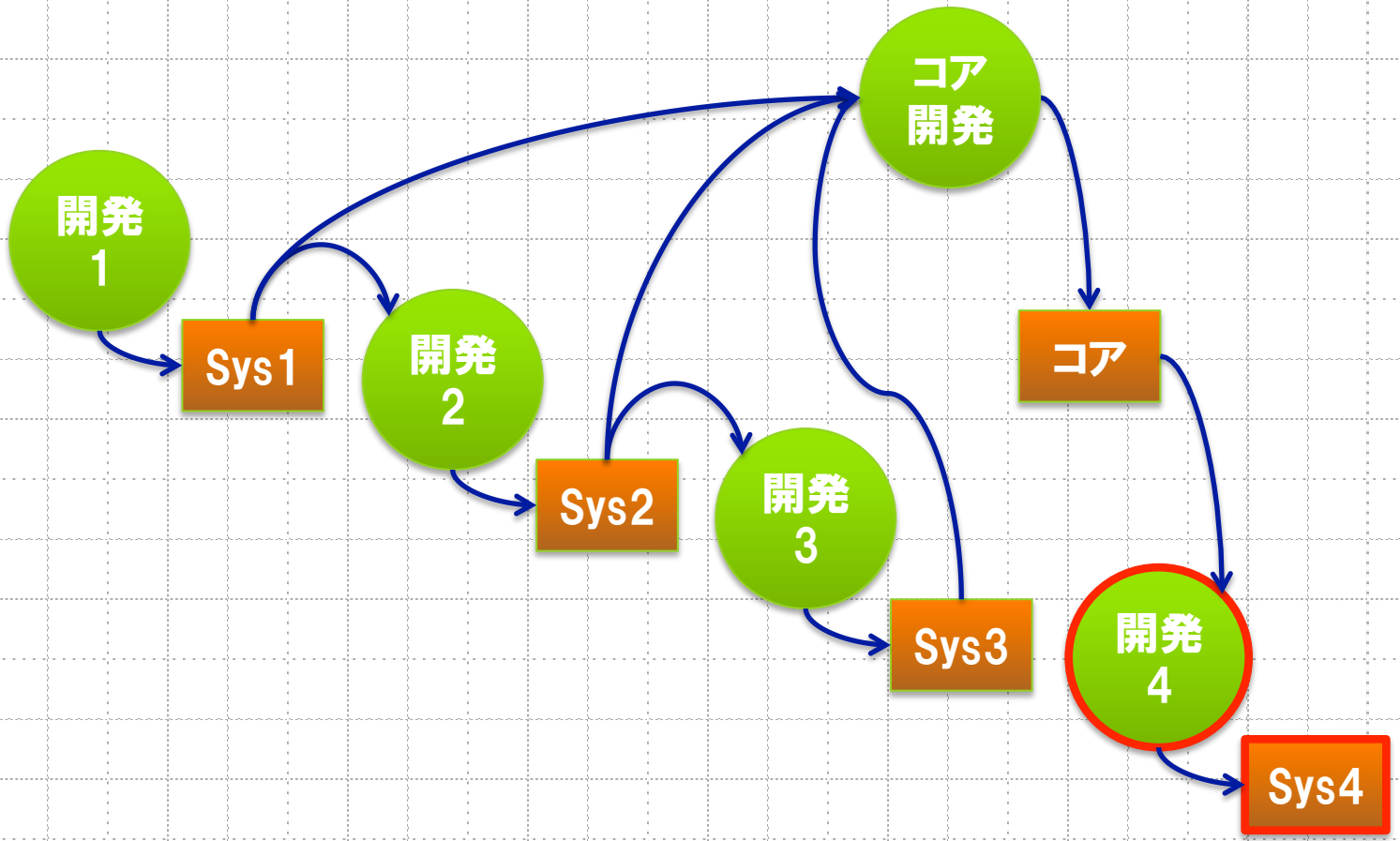
- 準備のコスト・期間: かかる(アプリケーション開発ごとに、結果に基づいて)

↓

- アプリケーション開発コスト・期間は小さい
- 事前準備式、都度事前対応式よりも環境変化への対応が容易
  - 新たなバリエーションを作った後にコア資産を拡充するため
- 包括的な分析を行なわないので、コア資産開発の手戻りがありうる
- コア資産開発の分だけ出荷までの期間が長くなりうる
- コア資産で常に過去システムまでカバーするには、コア資産の変更に規則/仕組みを設ける必要がある
- アプリケーション資産からコア資産拡充部分を抽出するので…
  - コア資産拡充の難易度がアプリケーションの内部品質に依存する　または
  - コア資産拡充が容易になるように予めアプリケーションの造りに制約をかける
- 例: Hewlett-Packard

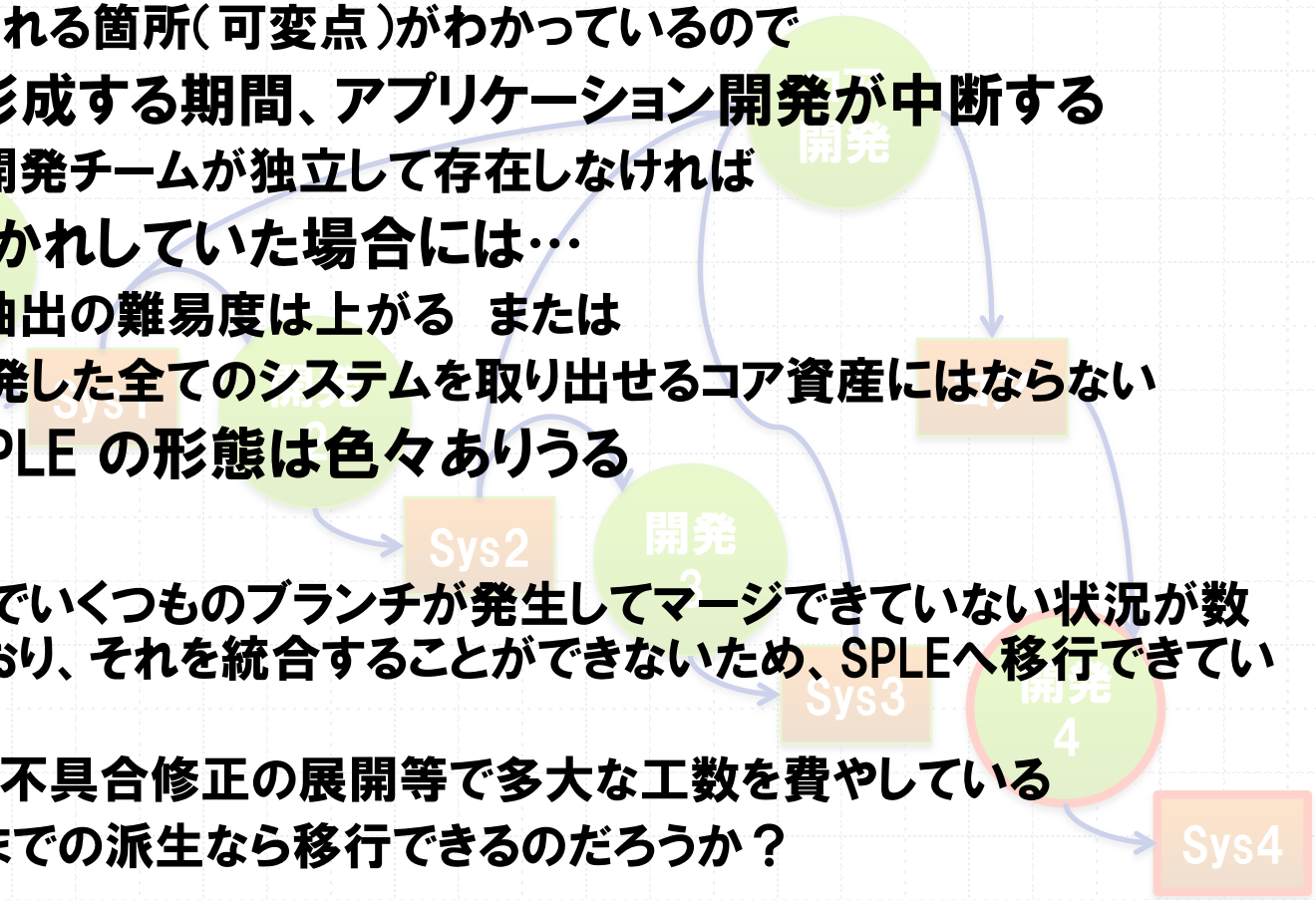


派生開発→SPLE 移行の様子(例)



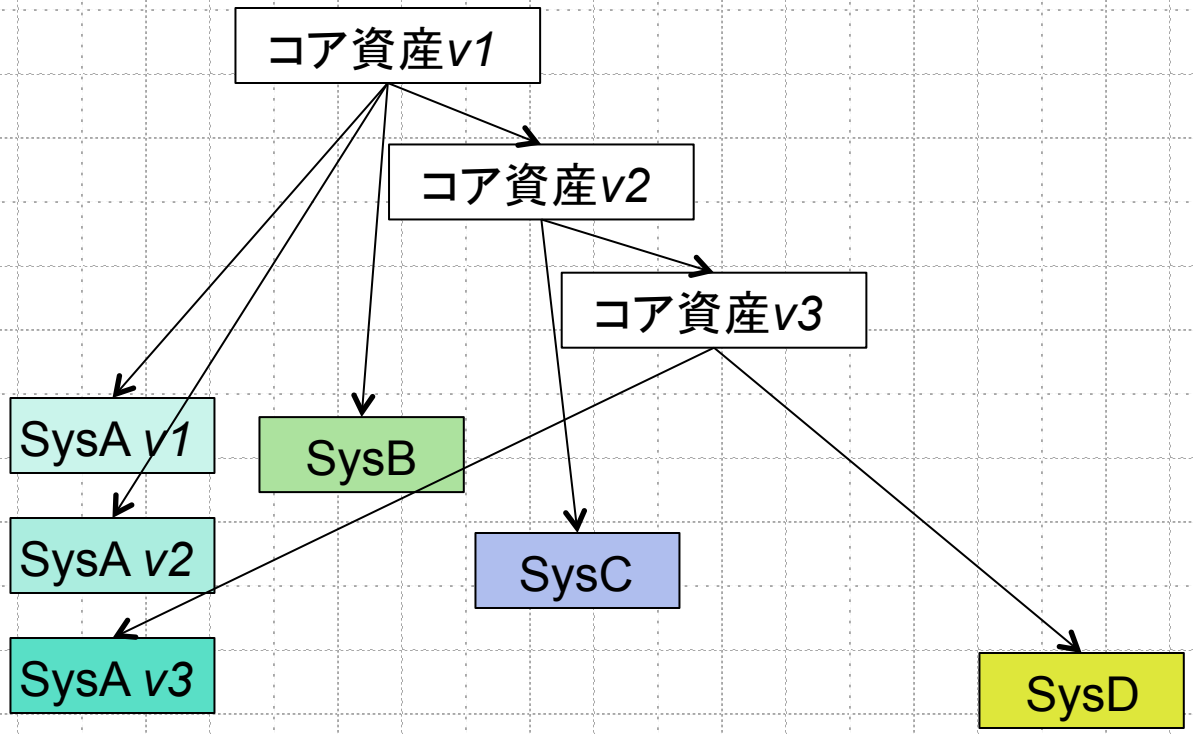
前ページの派生開発→SPLE 移行例の特徴

- 過去の派生開発での変更実績を基にしたコア資産の変更性が期待される
  - よく変更される箇所(可変点)がわかっているので
- コア資産を形成する期間、アプリケーション開発が中断する
  - コア資産開発チームが独立して存在しなければ
- 派生が枝分かれしていた場合には…
  - コア資産抽出の難易度は上がる　または
  - 過去に開発した全てのシステムを取り出せるコア資産にはならない
- 移行後の SPLE の形態は色々ありうる
- とある例:
  - 派生開発でいくつものブランチが発生してマージできていない状況が数年続いており、それを統合することができないため、SPLEへ移行できていない
  - そのため、不具合修正の展開等で多大な工数を費やしている
  - どの程度までの派生なら移行できるのだろうか？



コア資産初期開発ありの SPLE ―実際

- コア資産に対しては派生開発が行なわれる



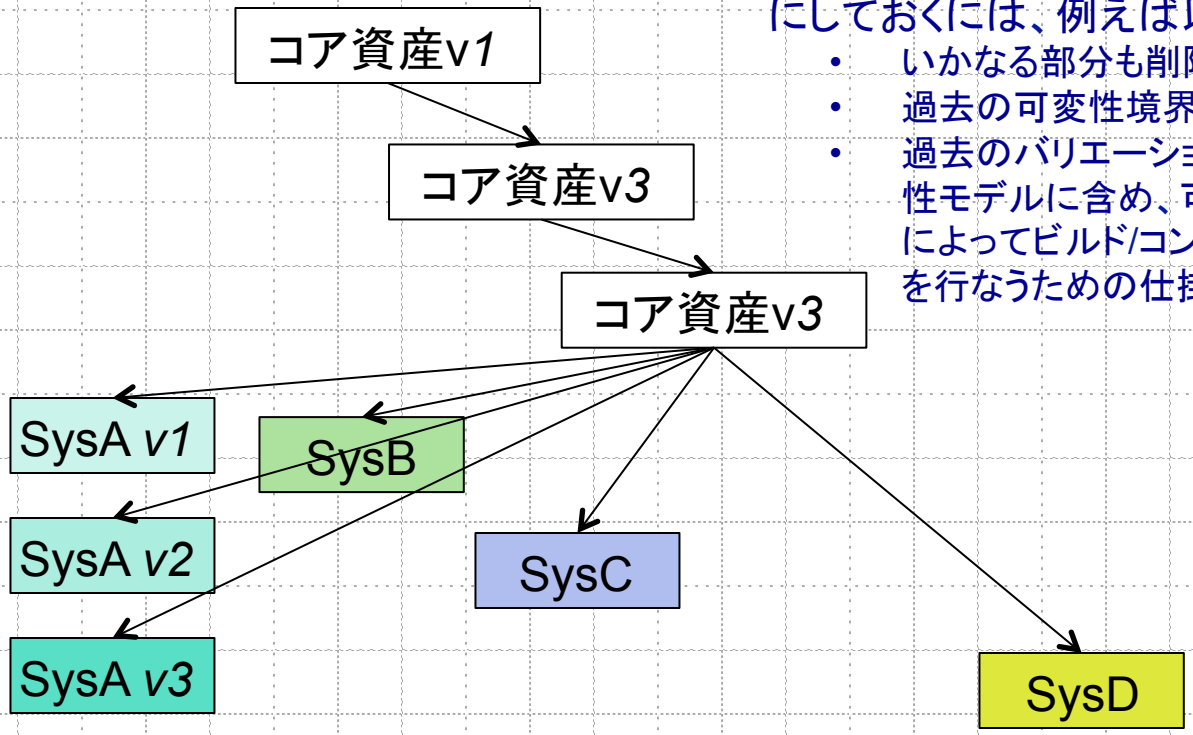
派生開発と SPLE の違い：元資産の更新

- SPLE でのコア資産保守発展ではバックワードコンパチビリティを持たせるが、派生開発では「資産」が別のシステムであるので新システムとの整合性は問題としない
  - 派生開発では、派生元にフィードバックはしない
  - SPLE ではコア資産にフィードバックする

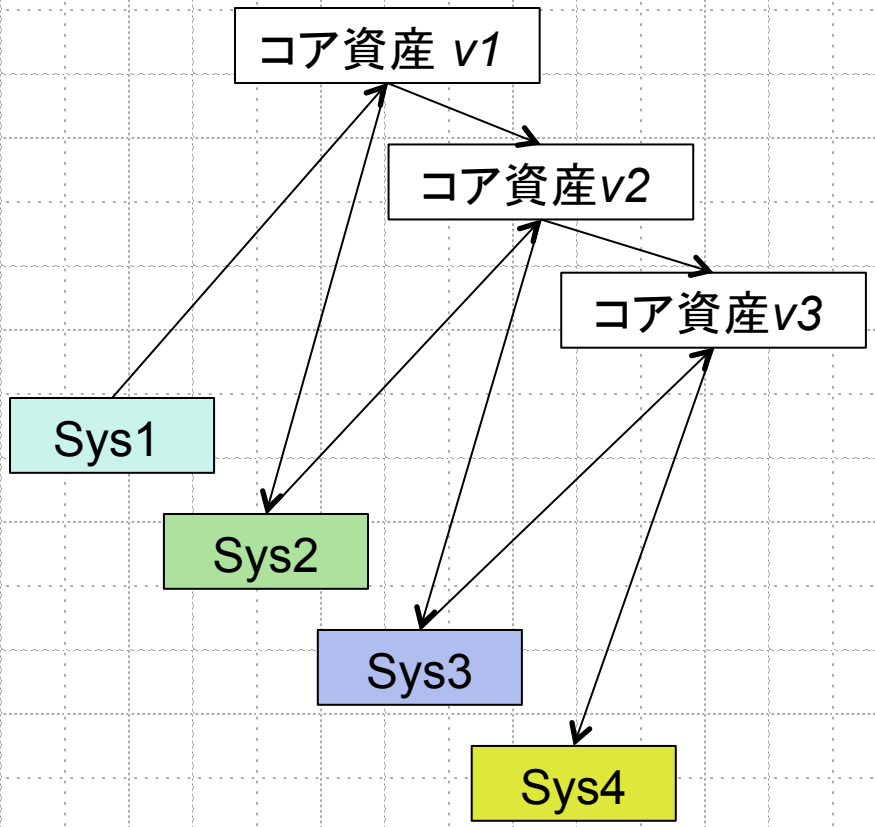
コア資産初期開発ありの SPLE ―目標

- 全てのシステムが最新のコア資産からビルド/コンフィギュア可能にしておく

- コア資産の改版を重ねてもそこから過去の版に基づいたシステムを得られるようにしておくには、例えば以下が必要
  - いかなる部分も削除しない
  - 過去の可変性境界を保持する
  - 過去のバリエーションも最新の可変性モデルに含め、可変性の取捨選択によってビルド/コンフィギュレーションを行なうための仕掛けを持つ

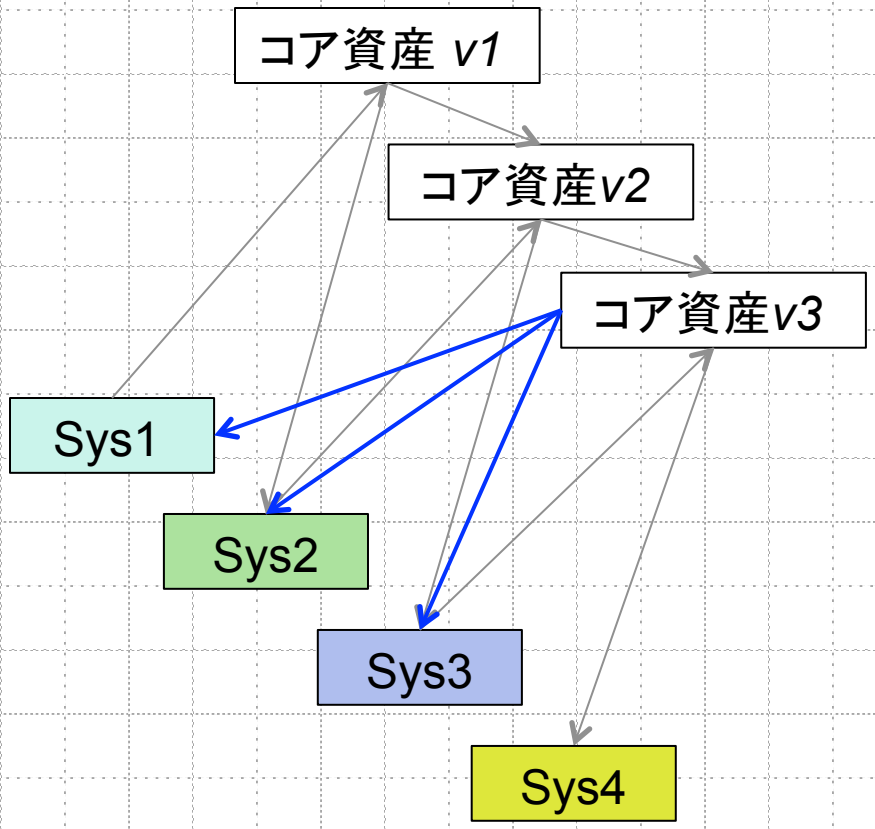


事後都度対応式 SPLE

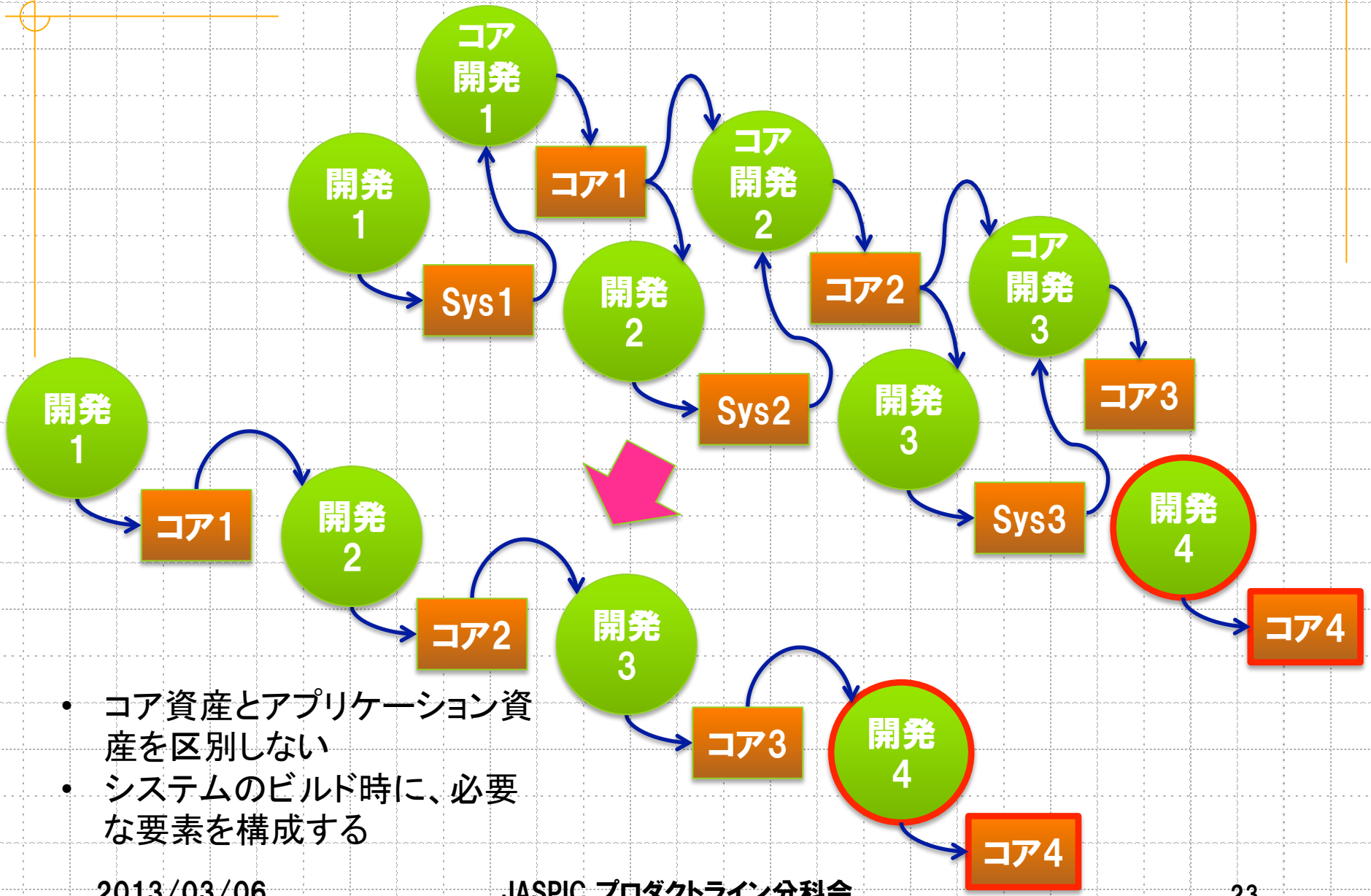


事後都度対応式 SPLE ―目標

- コア資産の最新版から過去のシステム全てが容易に再構築できるようにしておく



SPLE(統合・都度対応式)の様子



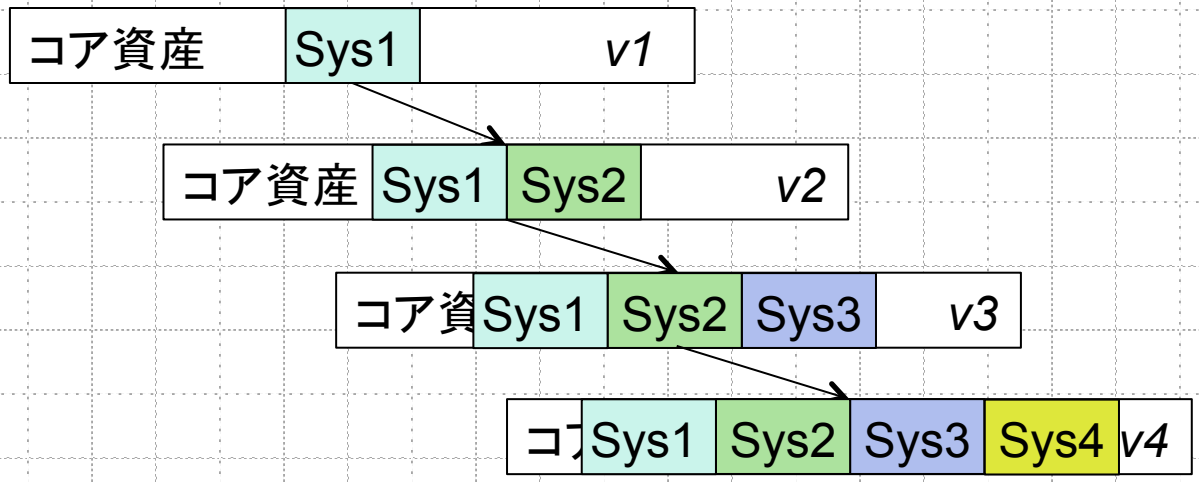
- コア資産とアプリケーション資産を区別しない
- システムのビルド時に、必要な要素を構成する

SPLE(統合・都度対応式)の特徴

- 準備のコスト・期間: 少々
  ↓
- アプリケーション開発コスト・期間は小さい
- 開発チームを二つに分けない
- コア資産から個々のシステムを容易に抽出する仕組みが必要
  - コア資産とアプリケーション資産を分けないため
- 事前準備式よりも環境変化への対応が容易
- 包括的な分析を行なわないので、コア資産開発の手戻りがありうる
- アプリケーションを抽出する仕組みに合わせて開発する分だけ出荷までの期間が長くなりうる
- コア資産で常に過去システムまでカバーするには、コア資産の変更に規則/仕組みを設ける必要がある
- 例: Salion, 以前の Hewlett-Packard

統合・都度対応式 SPLE

- コア資産の中に各アプリケーション資産の全てが含まれており、アプリケーション「開発」でそれを取り出す



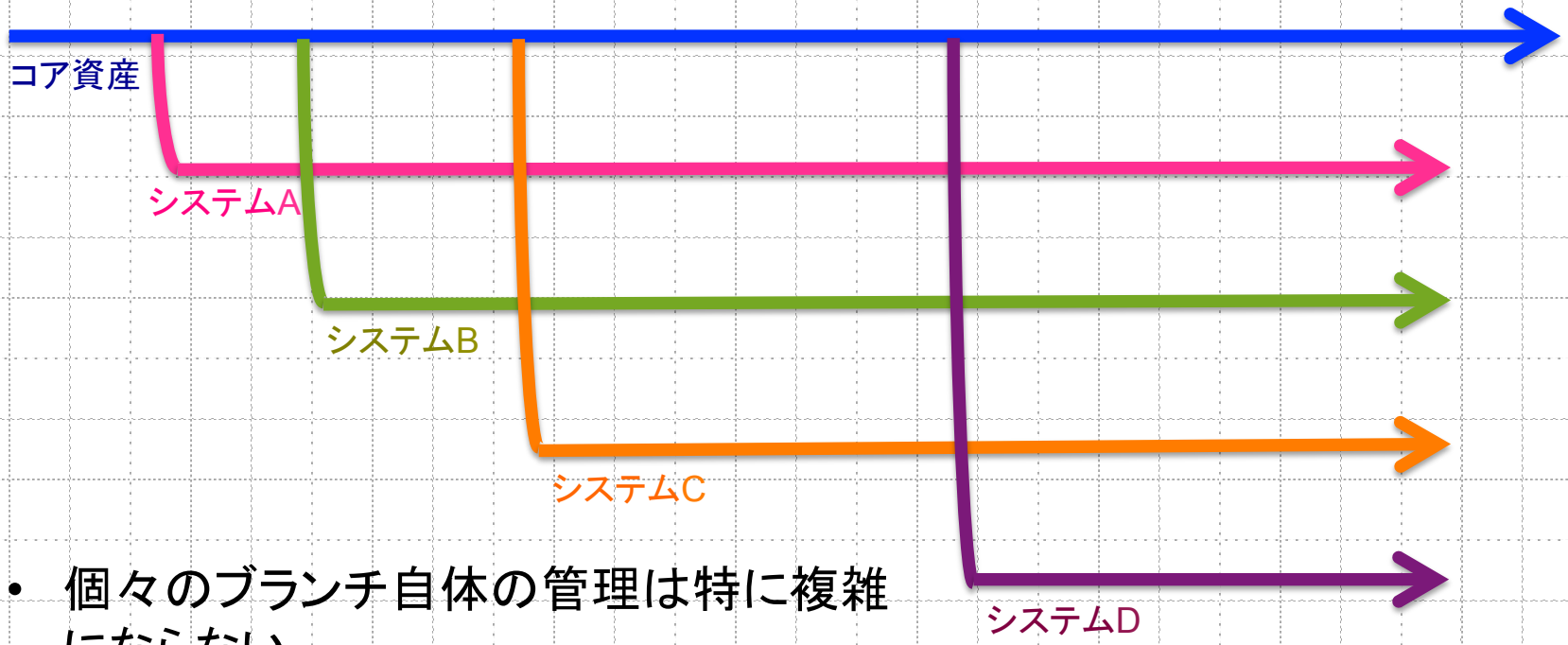
SPLE での構成管理

- コア資産がある
  - 構成管理が必要
- アプリケーション資産ができる
  - 構成管理が必要

↓

- トランク(コア資産)の進化(変更)は仕方ないが、ブランチ(アプリケーション資産)の枝分かれが増える一方
  - 派生開発と変わりなくなる
    - 枝分かれの枝分かれもできる？
  - 実際には何らかのフィードバックがコア資産にかかるが

アプリケーション資産のブランチ



- 個々のブランチ自体の管理は特に複雑にならない
- 似たような資産が複数できる
  - 不具合改修は大変
- この形ならコア資産に基づいて開発するより派生開発を行なう方が楽？

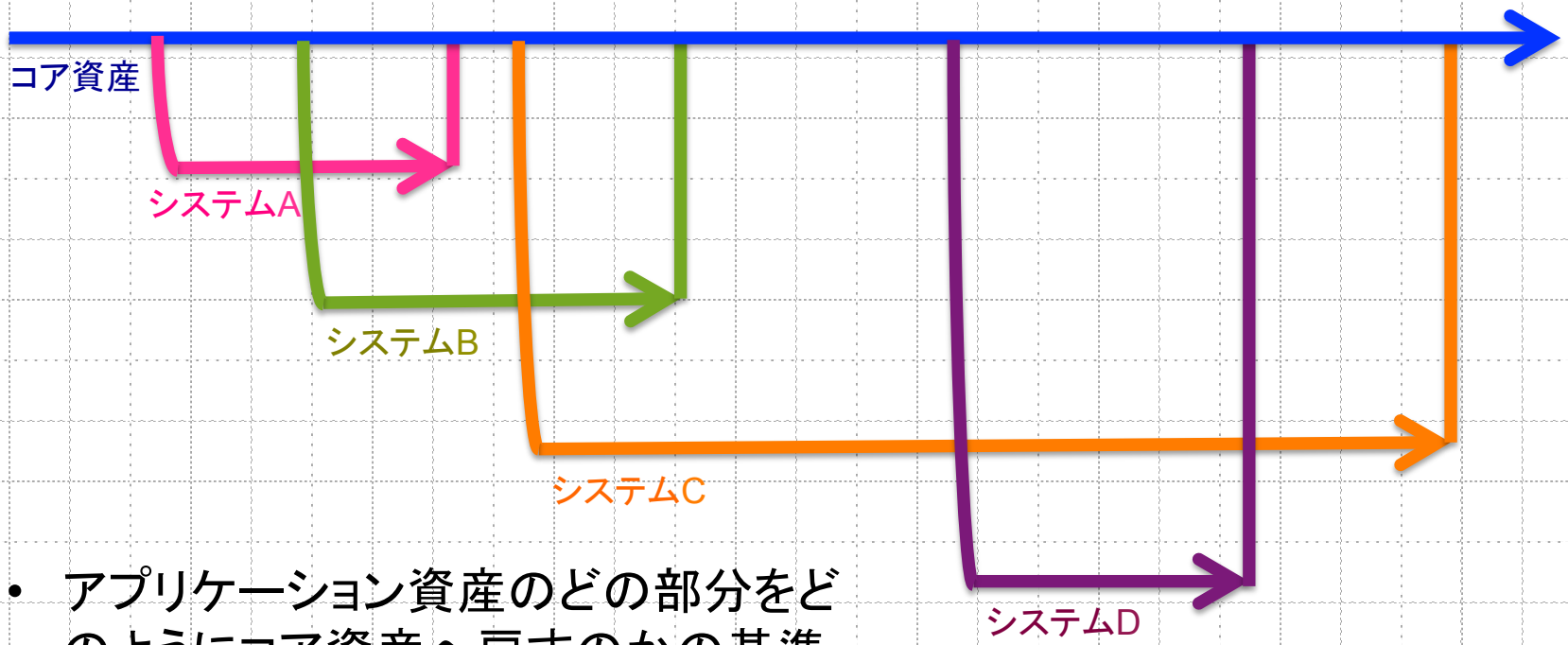
HP Owen Firmware Coop での構成管理

- コア資産がある
  - 構成管理が必要
- アプリケーション資産ができる
  - なるべくコア資産を直接変更する(適用可能対象を増やす)が、一部は一旦コピーする
    - 構成管理が必要
  - 但し適応改変の一部はコア資産に組み込まれ、それらはコア資産の構成管理下に戻される

↓

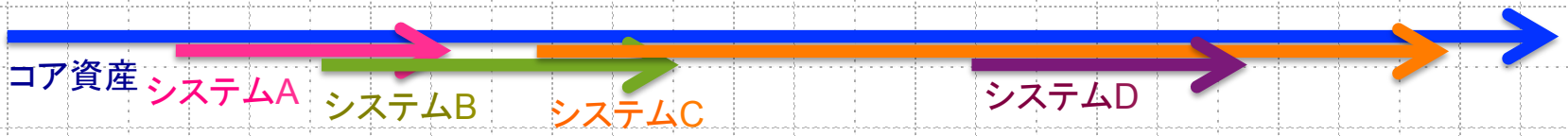
- アプリケーション資産の構成管理は branch 時から hard freeze までで一旦停止する
- 次ページ参照

コアおよびアプリケーション資産のブランチとマージ（都度事後対応式）



- アプリケーション資産のどの部分をどのようにコア資産へ戻すのかの基準の策定と実行が容易でない
  ※図のシステム D の資産を戻すのは特に難しい

アプリケーション向けにコアを直接編集（統合・都度対応式）



- システム間で何をマージし何を個別に保持しておくかの基準はやはり必要
- 並行開発が行なわれる場合、排他制御等の並行プロセス管理が必要

討論

討論のネタ

- 各方式や派生開発等からのSPLE移行に関する質問/感想はありますか？
- 派生開発をしている方は、SPLEに移行してみたいですか？
- SPLE に移行できない理由はありますか？
- 既に SPLE を実施している場合、あなたの組織では、どのパターンを採用していて、そこでの良かった点や困っている点は何ですか？
- ここであげた以外のパターンを採用しているケースをご存知でしたら、 紹介をお願いします。その方式の長所と短所は何ですか？
- 今後、採用したいと思ったパターンはどれですか？ その選択の理由は何ですか？
- 現状の構成管理でどのような構成管理ツールを使っていますか？ 助かっている点や、困っている点は何ですか？
- こんな機能がツールに欲しいと思っていることはありますか？
- 構成管理についての質問/コメントはありますか？
- その他