Integration as a Service
現状と課題

M2011MM041 小島 弘誉

概要

- 前回からの流れ
- 論文の概要
- シナリオ
- 提案されたシステムの全体像
  - Parameterized EAI Patterns
  - Process Guideline
  - Annotation for Provisioning
  - Deployable Artifacts
  - Provisioning Flows
- まとめと今後の課題
- 考察

前回からの流れと今回の狙い

- 前回のおさらい
  - 前回の目的: Integration as a Serviceの背景を明確化
  - 5つの連携アプローチと3つの連携方法論

- Hosted and extended ESB
- Message Queue, Brokers and Hubs
- Configured-based Platform
- Service Portfolio Approach
- Appliance-based

- 改良された伝統的な連携ツール
- クラウドホストの伝統的な連携ツール
- Integration-as-a-Service

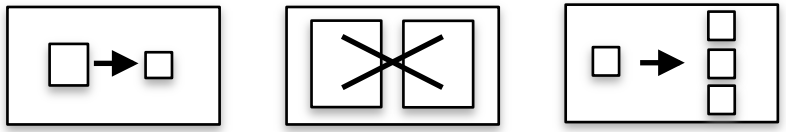
- 今回の狙い: 連携ツールをクラウドにデプロイは可能か
  - 論文: EAI as a Service
    Combining the Power of Executable EAI Patterns and SaaS
  - 参考文献: Defining Composite Configurable SaaS Application Packages
    Using SCA, Variability Descriptors and Multi-Tenancy Pattern

論文の概要

- 提案
  - EAI (Enterprise application integration) as a Service
  - 複雑な連携を実現するinfrastructure
- アプローチ
  - **EAI Pattern**

    統合パターンに共通するものを分類
    - ✓ 提案者: Gregor Hohpe
  - **Component Deploy Pattern**

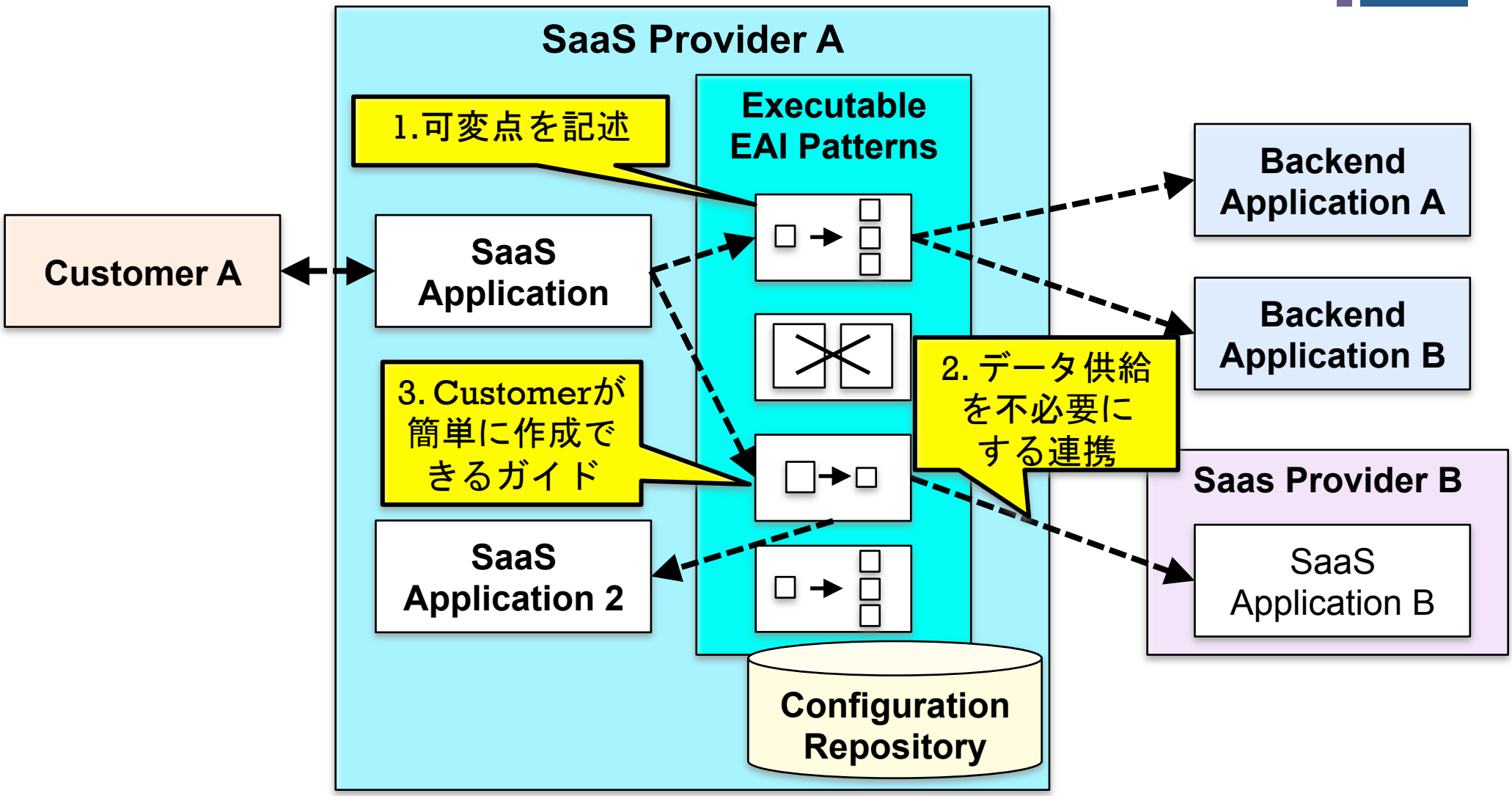
    Multi-tenancyのためにコンポーネントを分類
    1. single instance
    2. single configurable
    3. multi instance



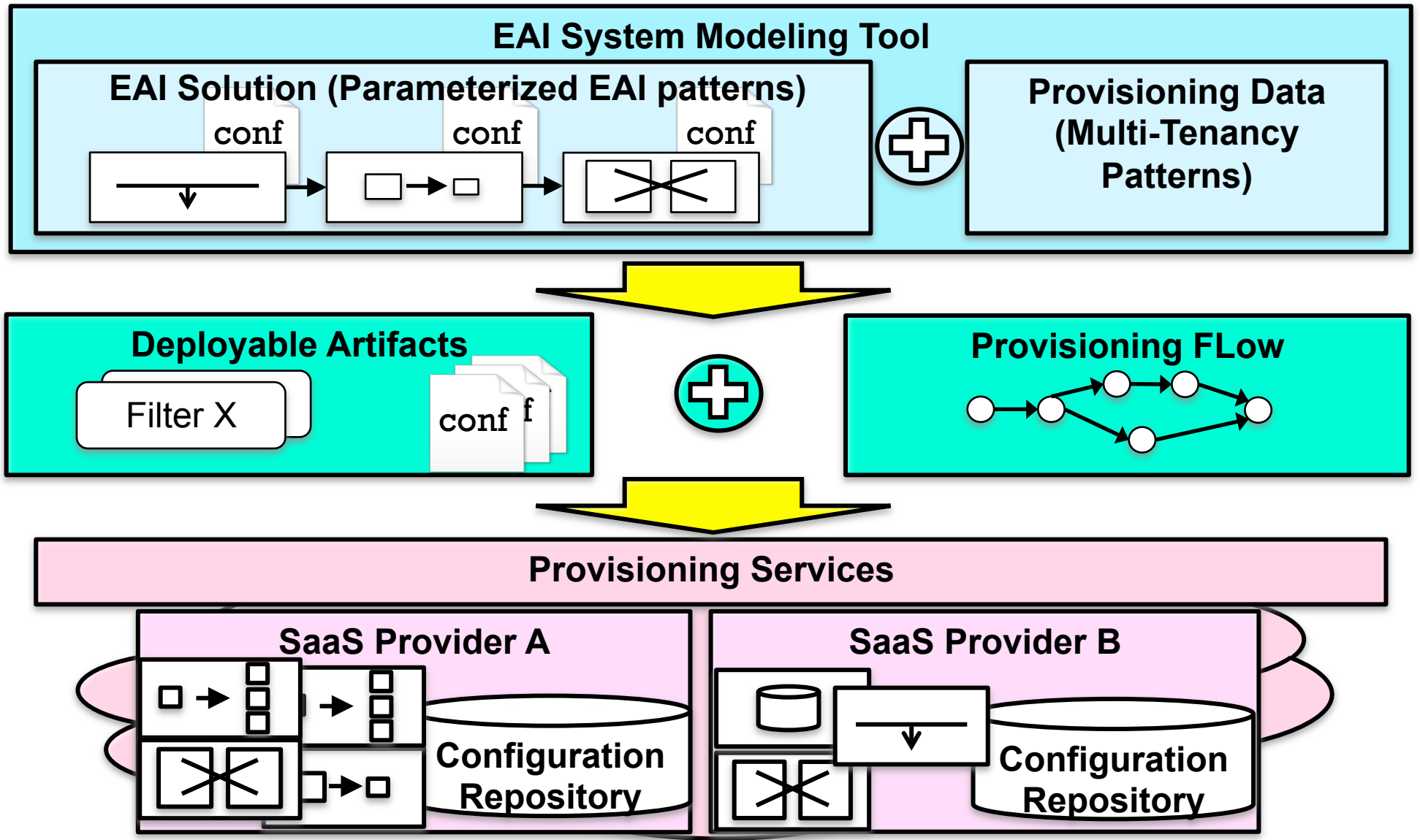
シナリオ

- SaaS導入企業: PerfectTravel
  - アプリケーションホスト経験なし
  - アウトソーシング
- アウトソーシング先: PerfectHosting
  - SaaSアプリケーションホストの専門企業
- 要求と制約
  1. Customer毎に連携のカスタマイズ実現
  2. CustomerのSaaSと他のシステム間でデータ供給を不必要に
  3. Customerはどのバックエンドシステムを連携させるか,インタフェースをいくつ持つかわからない

シナリオを実現するEAI pattern

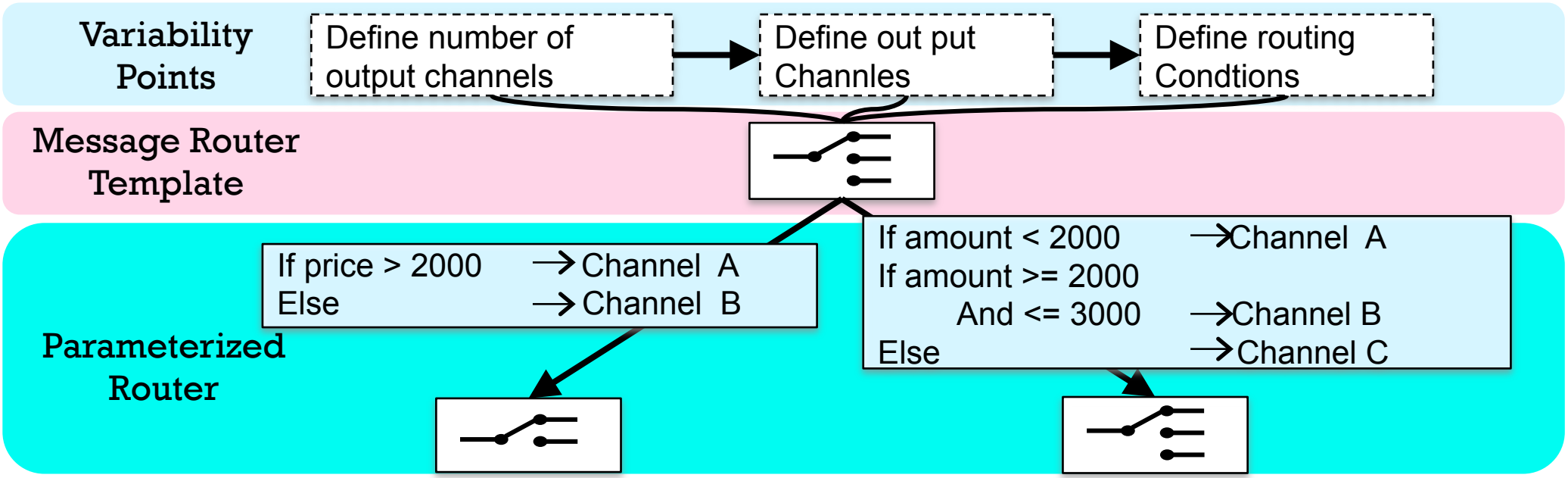


提案されたシステムの全体像



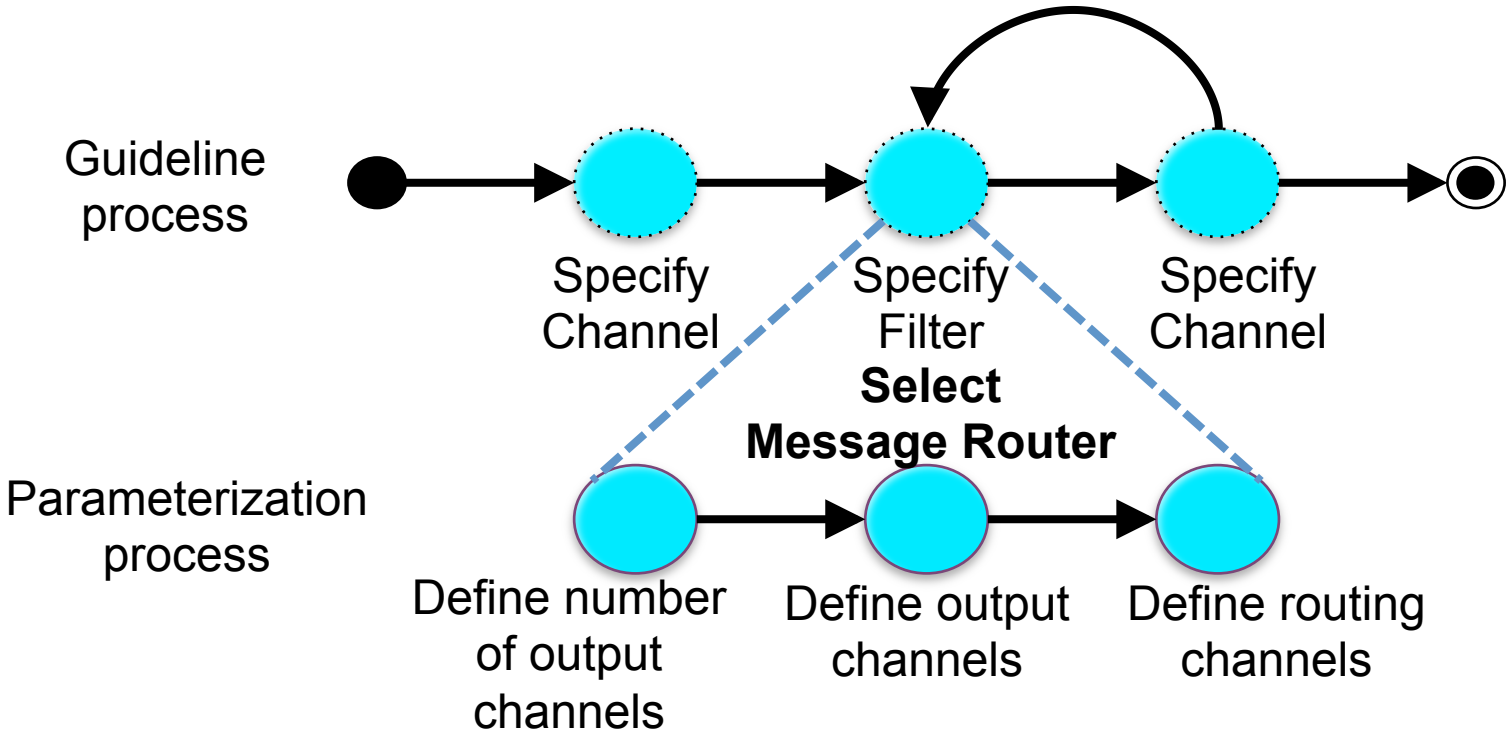
Parameterized EAI Patterns

- Validity Descriptor
  - EAI patternに可変点を記述 (feature treesと類似)
  - 制約を記述
  - 可変点のための選択肢
  - 依存関係
- Message Routerの例



Process Guideline

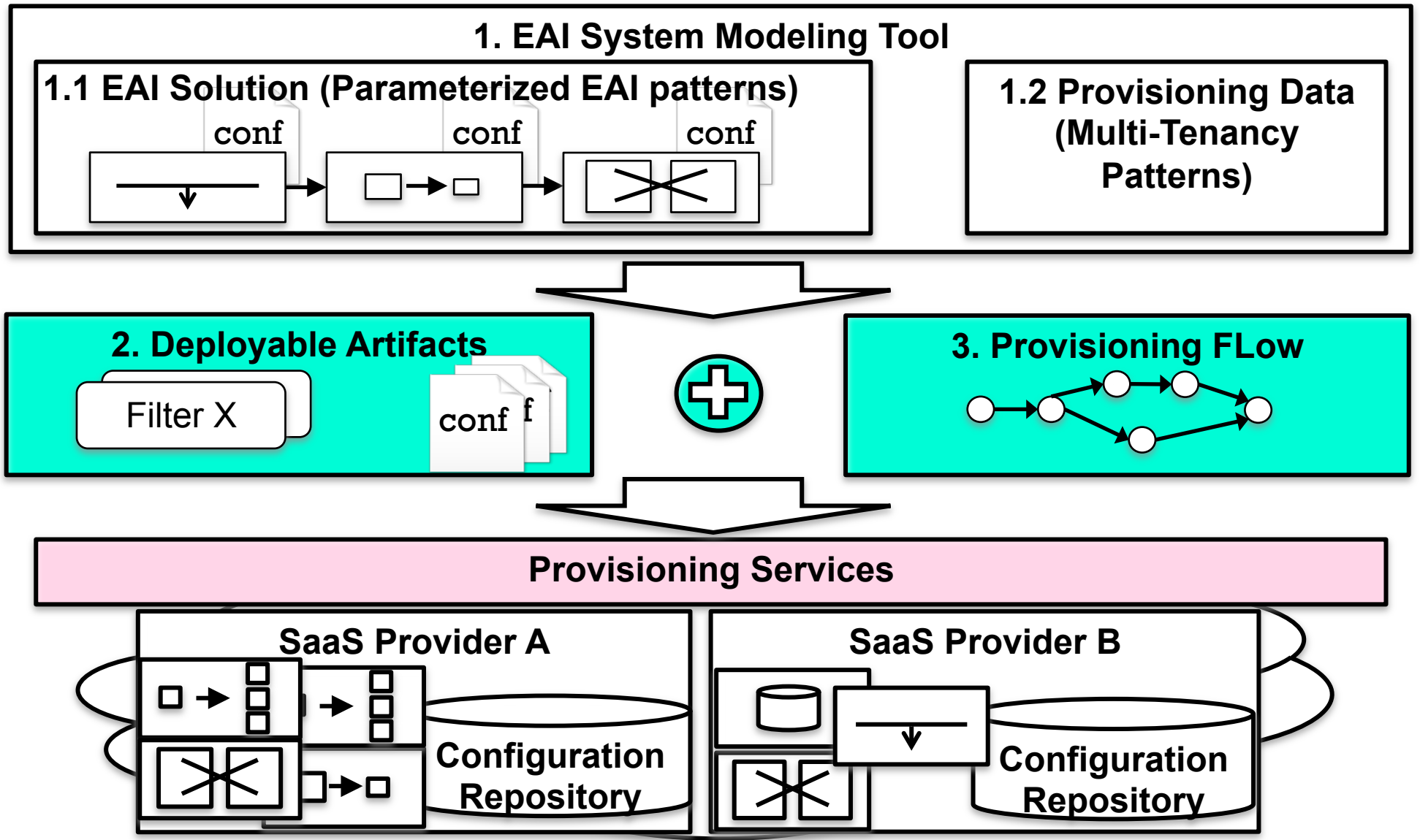
- 利用技術: Workflow
- 実装: WS-BPEL
- ガイドラインプロセスは抽象度が高い



Annotations for Provisioning

- Annotations for Provisioning
  - SaaS環境上にデプロイのためプロビジョニング情報が必須
  - 要求によりパターンは異なる
- EAI system model systemで付与する情報
  1. 選択したEAIパターン
  2. 各パターンのパラメータ
  3. どの構成でデプロイされるべきか

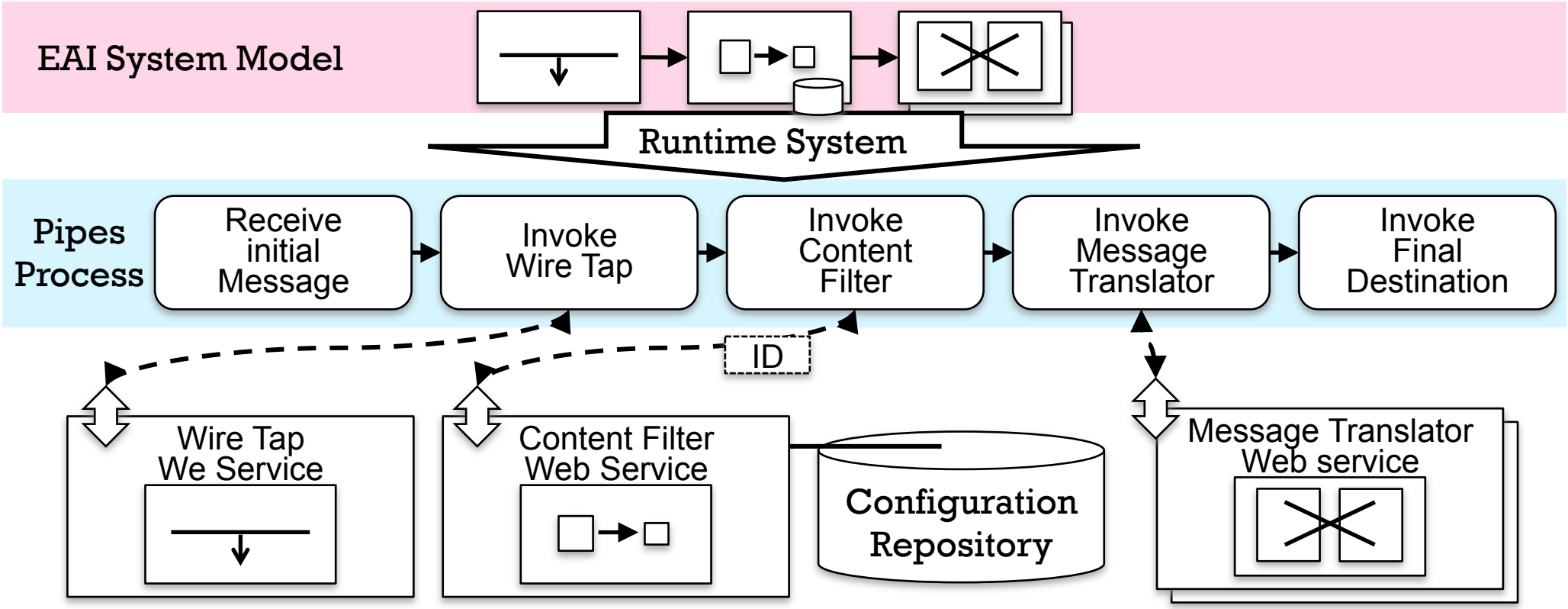
提案されたシステムの全体像



Deployable Artifact

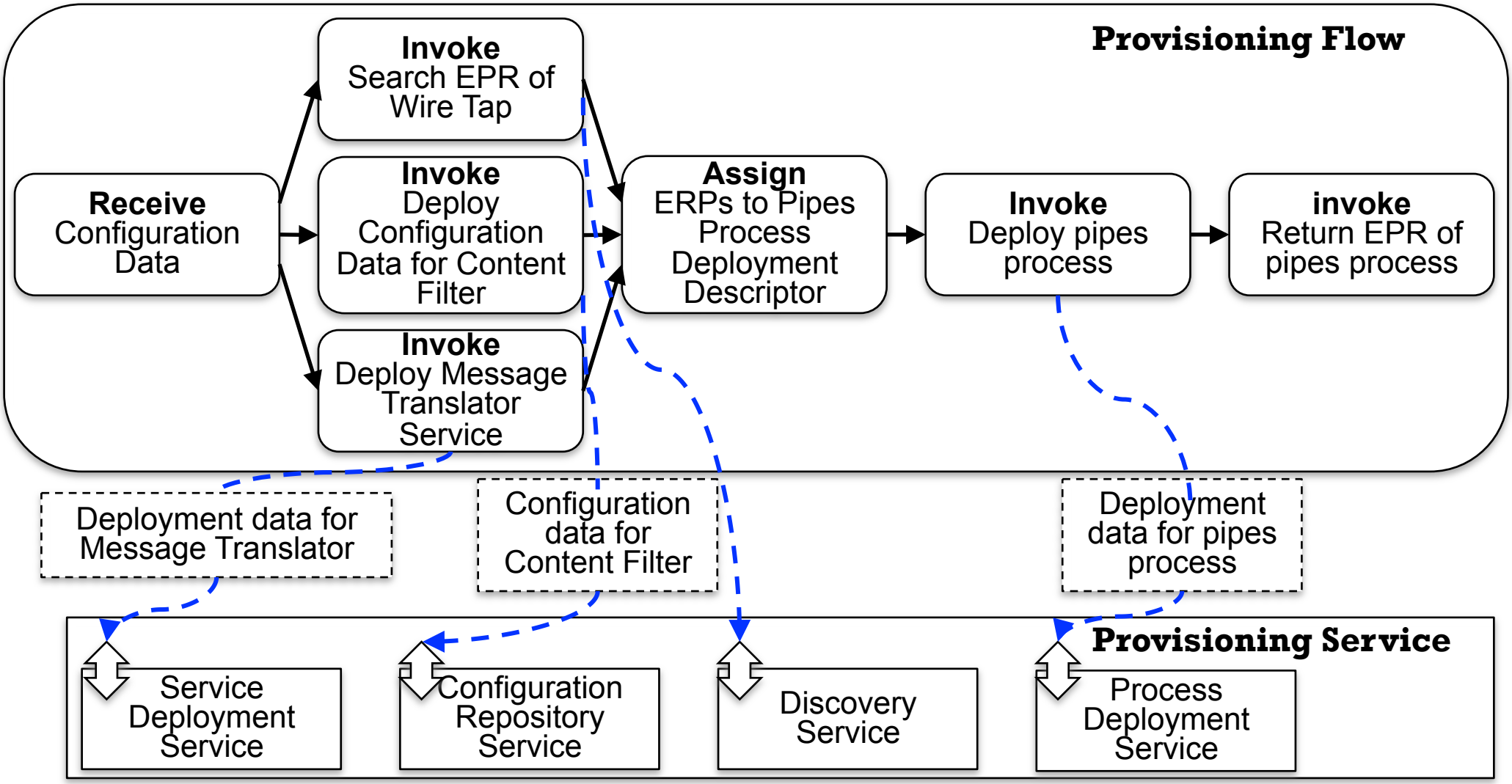
Artifactとは全体のフィルタ

- Artifact実行には以下のパラメータに依存
  1. Programming Language
  2. Multi-tenancy Deployment Pattern



Provisioning FlowsとProvisioning Service

Provisioning Flowsとはアクションの実行順と並列処理の記述



まとめと今後の課題

- 提案: EAI as a Service
  - 複雑なITインフラストラクチャのセットアップがない
  - EAI patternの利用
  - workflow driven approach
- 自動生成
  - 実行コードが生成されることを示した
  - multi-tenancyがコード生成にどのように影響を与えるか
- 今後の課題
  - 非機能要求のプロパティを含んだ連携 (可用性のような)
  - provisioning flowがどのような非機能要求の連携を生成できる

考察

- 連携ツールをクラウドにデプロイは簡単か
  - 今回の論文だけでは判断できない
  - 想定されるシナリオにVenderがいる(User, SaaS Provider, SaaS Vender)
- 技術の選択にSaaS特性が考えれていたか
  - EAI as a ServceはComponent Deploy Patternに基づく
- Integration as a Serviceを利用モデルをはっきりさせる
  - サービス上で開発がどの程度行えることが理想か
  - 連携シナリオを洗い出し，分類

今後の選択肢

- SaaSとコンポーネントの関係を調査
  - 論文: Defining Composite Configurable SaaS Application Packages Using SCA, Variability Descriptors and Multi-Tenancy Pattern
- 連携に必要なScalabilityとReliabilityの調査
  - 論文: ECB: Enterprise Cloud Bus based on WS-Notification and Cloud Queue Model
- Integration as a Serviceを利用するシナリオの明確化
  - 論文: Enterprise Cloud Service Architectuer
  - 連携シナリオを分類している論文を探す

参考文献

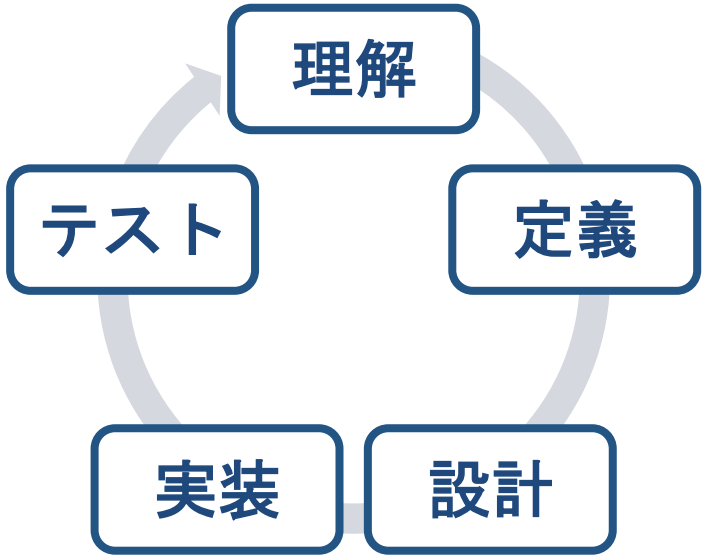
- SaaSの全て
- Architecture Strategies for Catching the Long Tail
  - http://msdn.microsoft.com/en-us/library/aa479069.aspx

ご清聴ありがとうございまいした

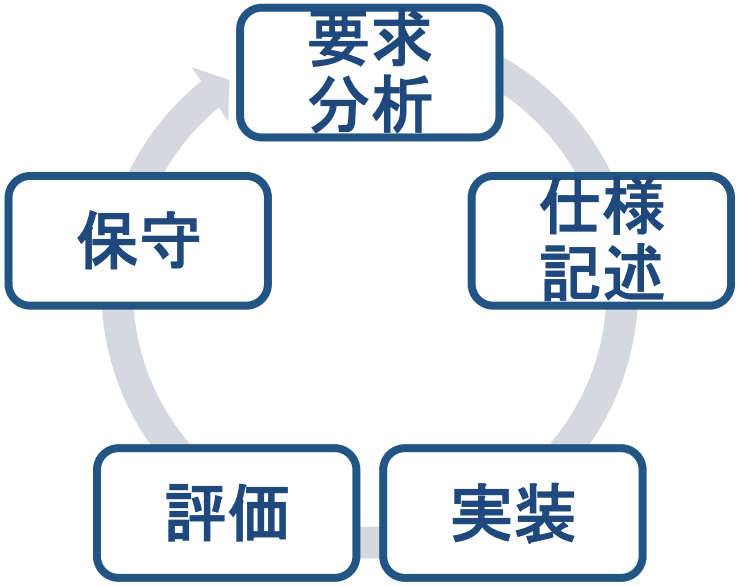
前回の質問

ソフトウェアライフサイクルと比較すると何が異なるか

インテグレーションライフサクル



ソフトウェアライフサクル



Provisioning Flows

Provisioning Flowsとはアクションの実行順と並列処理

- デプロイはpipeとfilterの数に依存