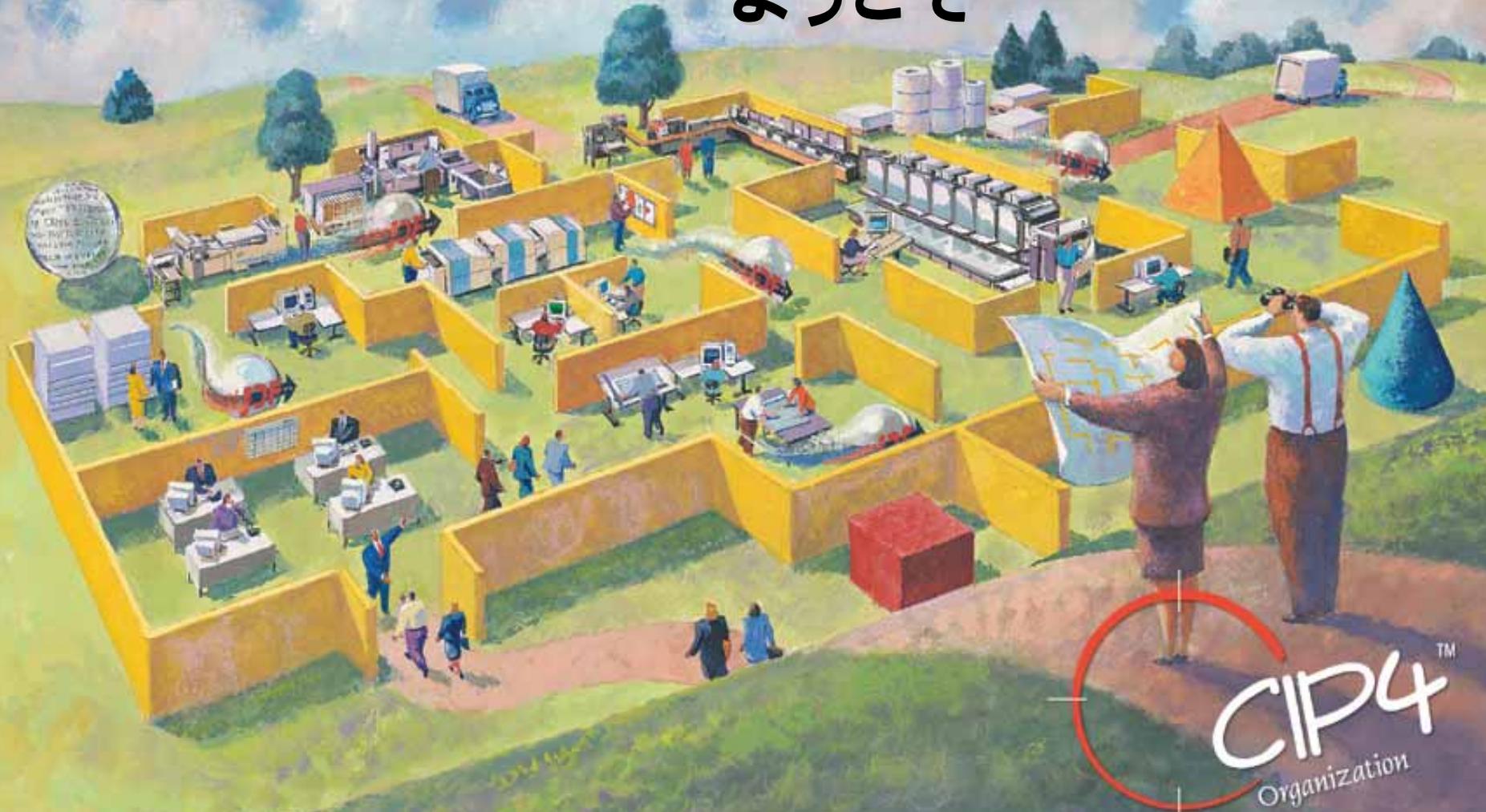




IGAS2007 (Tokyo) JDFチュートリアルへ ようこそ





チュートリアルセッション講師

- EFI社 Margaret Motamed
- Heidelberg社 Rainer Prosi





概要

- JDF構成概要
- JDFの現状





JDFに関して知る必要のないこと¹

```
<JDF ID="n1" Status="Waiting" Type="Product" Version="1.2"/>
<ResourcePool>
  <Layout Class="Parameter" ID="r112" Status="Unavailable">
    <Signature>
      <Sheet SurfaceContentsBox="0 0 1842.5197 1417.3228">
        <Surface Side="Front">
          <MarkObject CTM="1 0 0 1 42 66" Ord="0"/>
        </Surface>
      </Sheet>
    </Signature>
  </Layout>
</ResourcePool>
```

%PDF-1.3

%ãïó

```
1 0 obj <</Type /Page /Parent 9 0 R /Resources 3 0 R /Contents 2 0 R
/BleedBox [ 0 0 635 881 ] /TrimBox [ 0 0 627 873 ] /Thumb 37 0 R >>
endobj
```

```
2 0 obj<</Length 698 /Filter /FlateDecode>>
```

stream

```
H%ãSMk 1ÿ%p/ :|†q,Ëöxzk6 0H;î,, B(a°ûêîB³ iÿ¼'â™ @sï!$büPÓ³4úm0ŽÀx,œ=
```



1 - ただし、JDF開発者は除く





JDFとは(1)

- JDFはグラフィックアート業界のジョブチケット
データ交換フォーマット仕様
 - JDFはアプリケーションやシステム ではない!
- JDFは XMLベース
- コンテンツは埋め込み型ではなく、参照型
- JDFは拡張可能
- JDFは本来下記の意味構造に基づく:
 - Adobe PJTF
 - CIP3 PPF
- JDFフレームワークの決定
= JDFジョブ定義 + JMFメッセージ + JDF機能 + ICSドキュメント





JDF とは(2)

- 目的と必要条件 -

• なぜJDF?

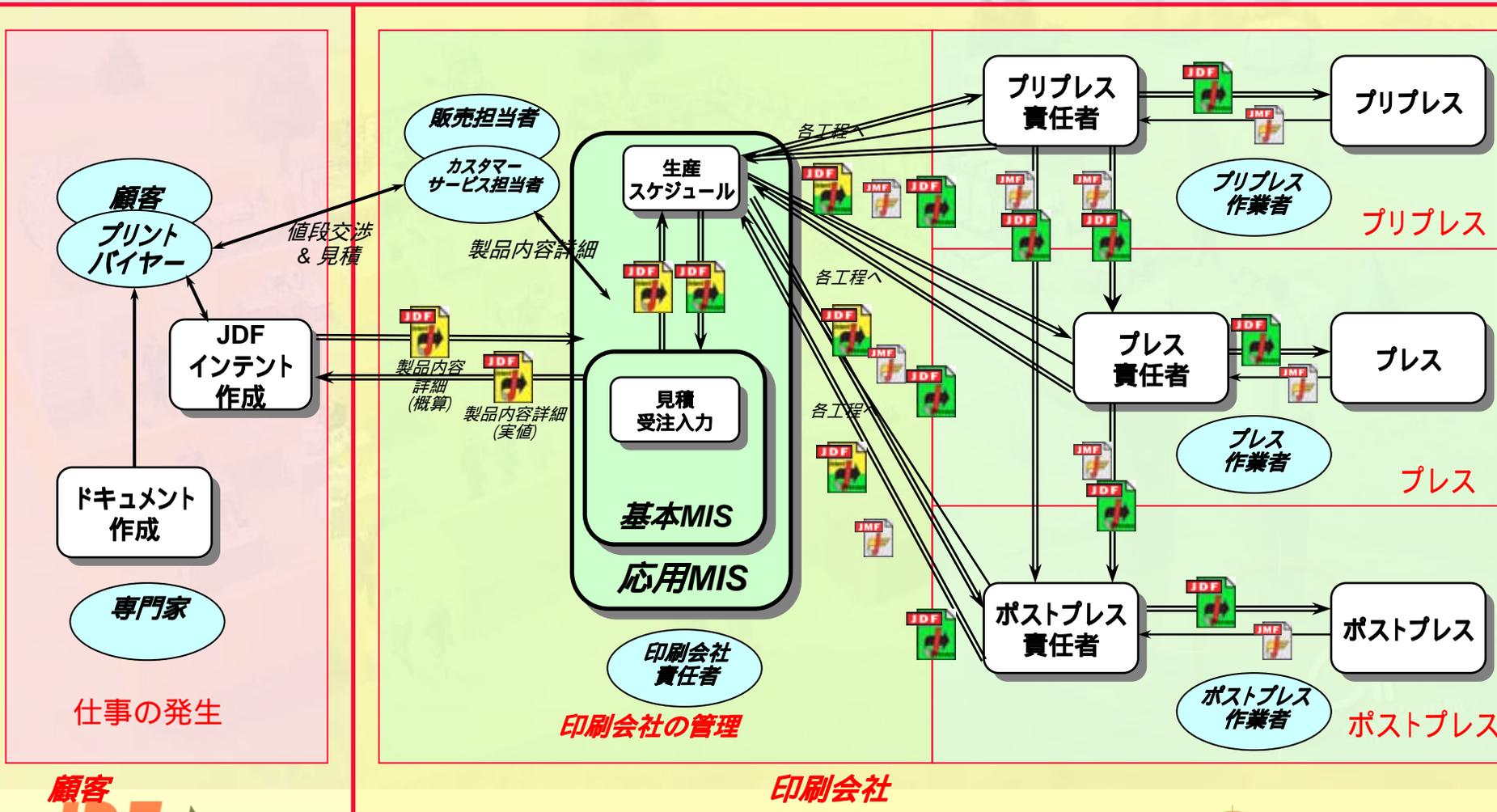
- 工程自動化により効率アップ
- デジタル情報によりミスを低減(紙ベースでの作業と比較)
- 情報に基づく決定が可能
- 紙ベースの管理ワークフローとデジタルコンテンツのワークフローを併用
 - 改善の余地有り

• JDFでできること

- 下記の点からグラフィックアート業界のジョブを確立:
 - テクニカルアプリケーション
 - 管理情報システム
 - 顧客
- 「川上」から「川下」までのジョブに関するデータを収集
 - ビジネスデータ
 - テクニカルデータ

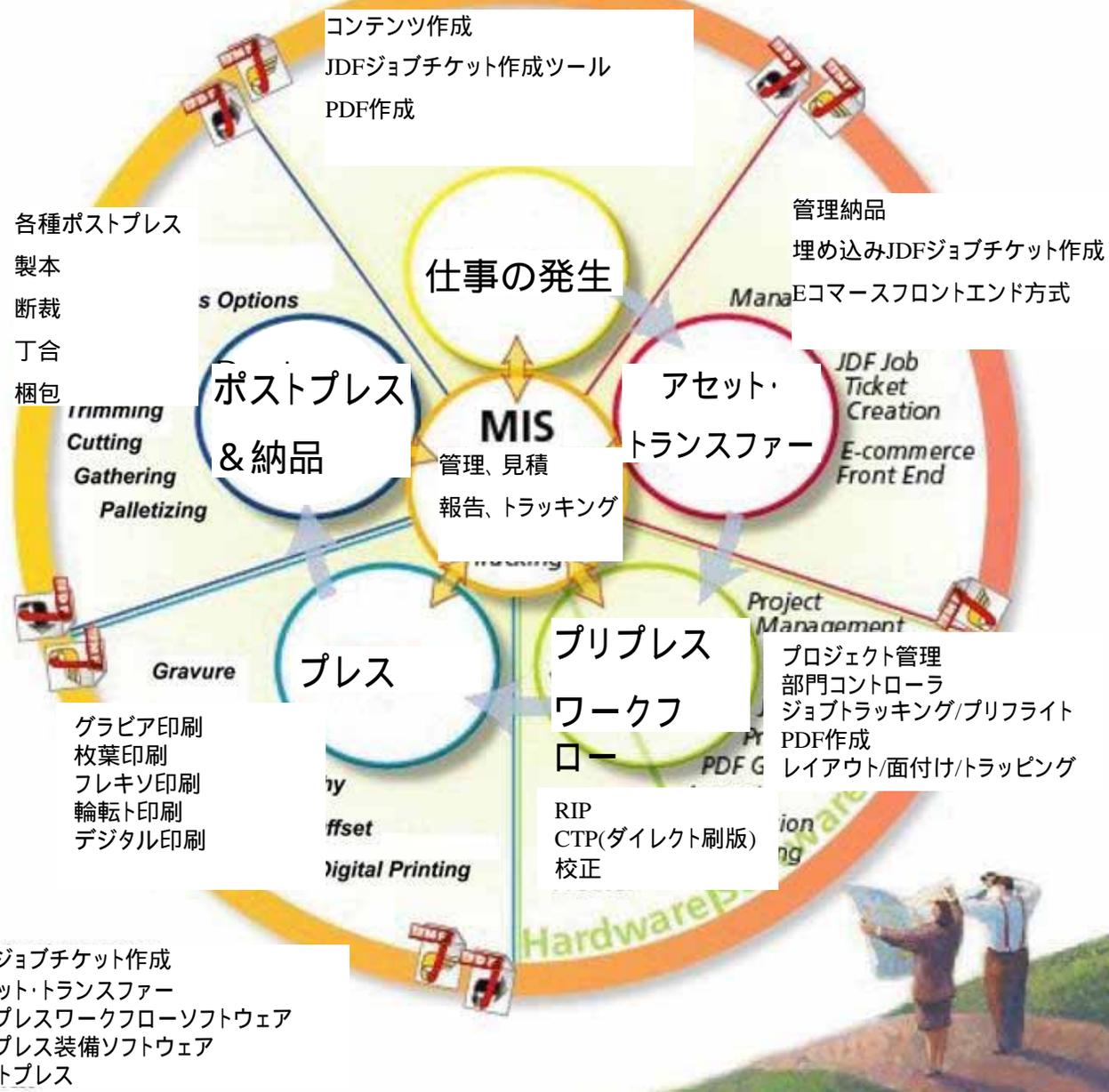


CIP4 リファレンスモデル





JDF ワークフロー





MISにおけるJDF



- 管理
 - 全ワークフロー工程に指示を伝達
- ジョブトラッキング
 - ジョブの追跡 (複数デバイスに配信可能)
 - デバイスの追跡 (複数ジョブを処理可能)
- 原価計算
 - 作業実績ベースの原価計算
 - JMF
 - Audits (JDFファイル内の実績データを記録する要素)
- 見積
 - 前回のジョブの原価計算フィードバックをベースに見積
- レポート
 - 複数ジョブとデバイスの統計データレポートを作成



ジョブ発生時におけるJDF



- ページ内容決定
 - 表紙
 - 本PDFデータ3ページ目が冊子本体の17ページにあたる
- 製品仕様決定
 - 使用媒体(用紙)
 - 製本方法
 - 色
- 制作作業の所要時間を算出
- 納期決定



アセット・トランスファーにおけるJDF (データ送信)



- 受信コンテンツファイルをユーザジョブへ
 - ファイル名規則の変更
 - 本PDFデータ3ページ目が冊子本体の17ページ
 - 雑誌Yの広告X
- プリフライト情報の維持
 - プリフライト概要 (確認事項)
 - プリフライト報告 (結果)
 - プリフライト概要と文書ドキュメントの変更



プリプレスワークフローにおけるJDF



- ジョブ進捗状況の追跡
 - 全ページの承認
 - 全版の用意
 - ...
- 色指定
- 校正
- スケジュール
- (請求用)作業時間記録
- 紙面レイアウト決定 – 面付け



プレスにおけるJDF



- ジョブ進捗状況の追跡
 - 全用紙の印刷
- 使用色分解の決定
- コーティング決定
- 媒体(用紙)決定
 - 供給印刷会社、供給出版会社
- 生産量、消費量の追跡と決定
 - 損紙枚数
 - 過剰生産枚数
- スケジュール





ポストプレスにおけるJDF



- ジョブ進捗状況の追跡
 - 折り完了
 - 綴じ完了
- 折り方法の決定
- 綴じ方法の決定
- 生産量、消費量の追跡と決定
 - 不良品量
 - 過剰生産量
- スケジュール



納品におけるJDF



- ジョブ進捗状況の追跡
 - 納入製品
- 梱包の決定
 - パレット
 - ダンボール
 - 箱
- 生産量、消費量の追跡と決定
 - 複数の製品を異なる組み合わせで複数の納品場所に納める場合にも対応できる
 - 各納品物の把握
- スケジュール



MISとは？

ソフトウェア関連略語:

- MIS: 管理情報システム
- ERP: 企業資源計画
- MES: 製造遂行システム
- PPS: 生産計画システム
- ...
- 印刷MISシステムは上記を網羅
 - 印刷業務に特化
 - 経営者に情報を提供



MISとは？

- 「MIS」とは(おそらく)誤った表現
 - MISとは1つの役割であって、必ずしも1つのシステムとは限らない
 - 主要業務:
 - 顧客への見積書提示
 - 顧客からの受注入力の処理
 - 生産ジョブとジョブチケットの作成
 - 生産計画を立てる
 - マテリアル(材料)・ロジスティクス(在庫、購買)の処理
 - 進捗追跡および原価計算の実行
 - 請求書の作成・発送、会計



MIS導入前

- すべて手作業
- 紙ベースの作業指示
- 生産情報の提供は紙ベース
- 作業完了時に書類を保管



JDF導入前

- MISがジョブに関する情報を管理、追跡
- 従来と同様、紙ベースのジョブチケットが主流
- 生産情報の提供は紙ベース
- リアルタイムに原価計算情報を把握するためには、専用のデータ収集端末が必要
- 作業完了時に書類を保管
- MISが保管、管理する印刷会社データ収集によりデータ掌握
- MISは、生産システムが管理、追跡した情報を把握できないことも多い



比較: MIS vs. 統合システム

- 2つの導入戦略

- MIS主導型

- MISは各デバイスを直接コントロール

- 統合システム主導型

- MISは各デバイスのコントロールを統合システムへ依頼
- 統合システムが各デバイスをコントロール

- 結論

- MISに生産コントロール機能を追加するか

- 生産コントロールシステムにMISの機能を追加するか



MIS主導の場合

- 長所:
 - 原材料の管理 (在庫)
 - 会計分析の即時対応
 - ヒューマンリソースの管理と統合管理できる
 - 事後計算 (精算見積) 用のデータ収集
- 短所:
 - MISは(デバイスを制御するために必要となる) 詳細な仕様情報まではわからない
 - 生産ワークフロー自体のコントロール不可



MIS中心のJDFワークフロー

- MISは既知のデータをJDF対応生産システムへ送信
- JDF対応生産システムはJDFチケットに更に多くの情報を追加
 - 生産パラメータ
 - 原価計算データ
 - 一般情報
- JDFチケットがMISへ戻る
- スムーズな情報伝達を確立



統合システム中心のワークフロー

- 長所:
 - 生産ワークフローを自動管理
 - 面付け処理の統合
 - 生産情報の統合
- 短所:
 - 会計分析の即時対応が困難



JDFのデータ表現

- XMLを利用
- URI/URLを使用した外部参照
- データ型定義にXMLスキーマを利用
- XML名前空間を使用した拡張性

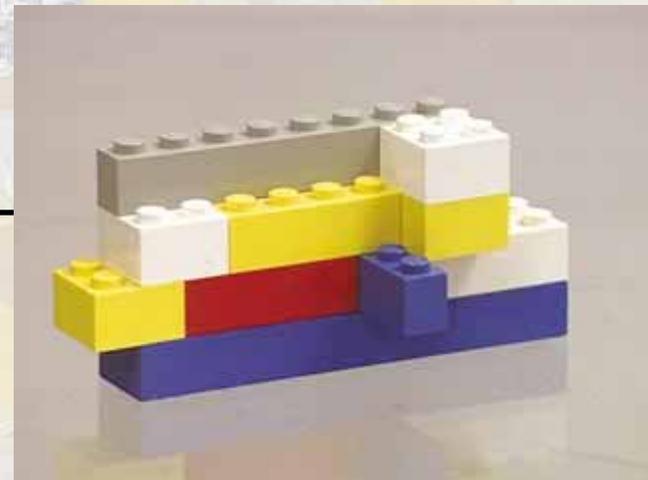
```
<JDF ID="N1" Type="Product" Status="Waiting" Version="1.3">  
  <ResourcePool>  
    <NodeInfo Class="Parameter" Status="Available" ID="Link0001" />  
    <SomeInputResource ID="Link0002" Class="Parameter"  
Status="Available"/>  
  </ResourcePool>  
  <ResourceLinkPool>  
    <NodeInfoLink rRef="Link0001" Usage="Input" />  
    <SomeInputResourceLink rRef="Link0002" Usage="Input" />  
</JDF>
```





JDFフレームワークの構築

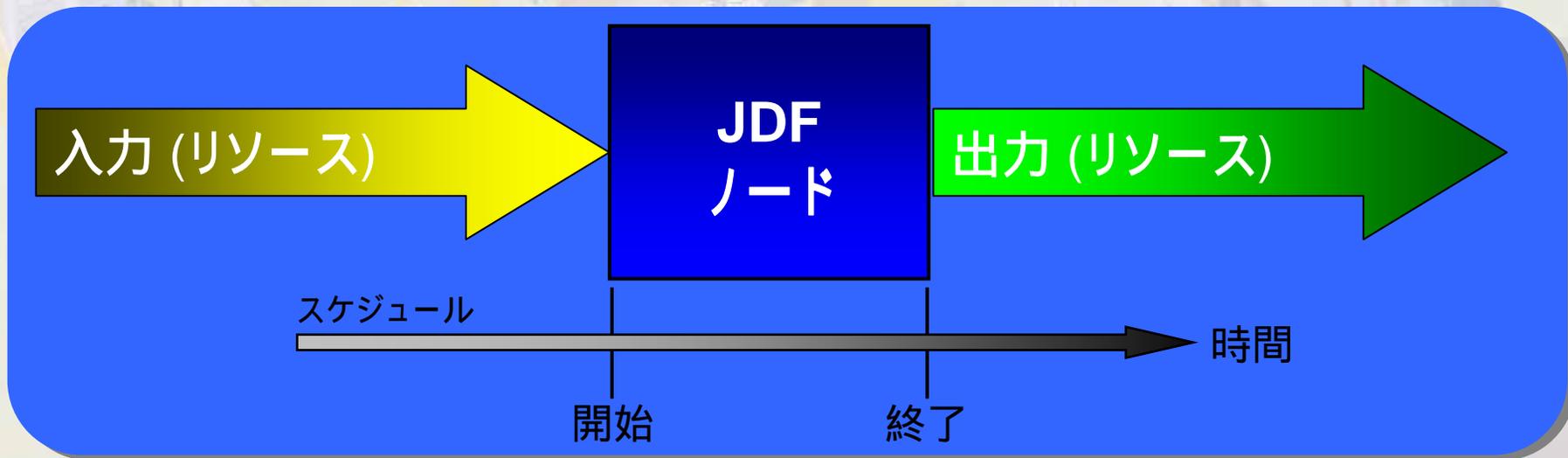
- JDFノード
 - 工程、工程グループ、グレイボックス、製品仕様の記述
 - JDFリソース
 - パラメータセット、物理エンティティ(実態)記述
 - JDFリソースリンク
 - JDFノードとJDFリソース間のリンク
-
- JDF機能記述
 - JDFインターフェース制限
 - JMFメッセージ
 - リアルタイムデータ交換フォーマット
 - ICSドキュメント
 - 各目的に合わせたJDF制限仕様





JDFノード

- 包括的な工程の記述
 - JDFの「動詞」
 - 指定の時間に実行
 - スケジュールや作業実績の記録場所
- 1つのJDFノードタイプが製品仕様と製造工程の両方に対応
- ジョブルートノードのおおまかな製品仕様
- 具体的な「工程グループ」ノードか「グレイボックス」ノード
- リーフノードの各詳細工程または統合した工程

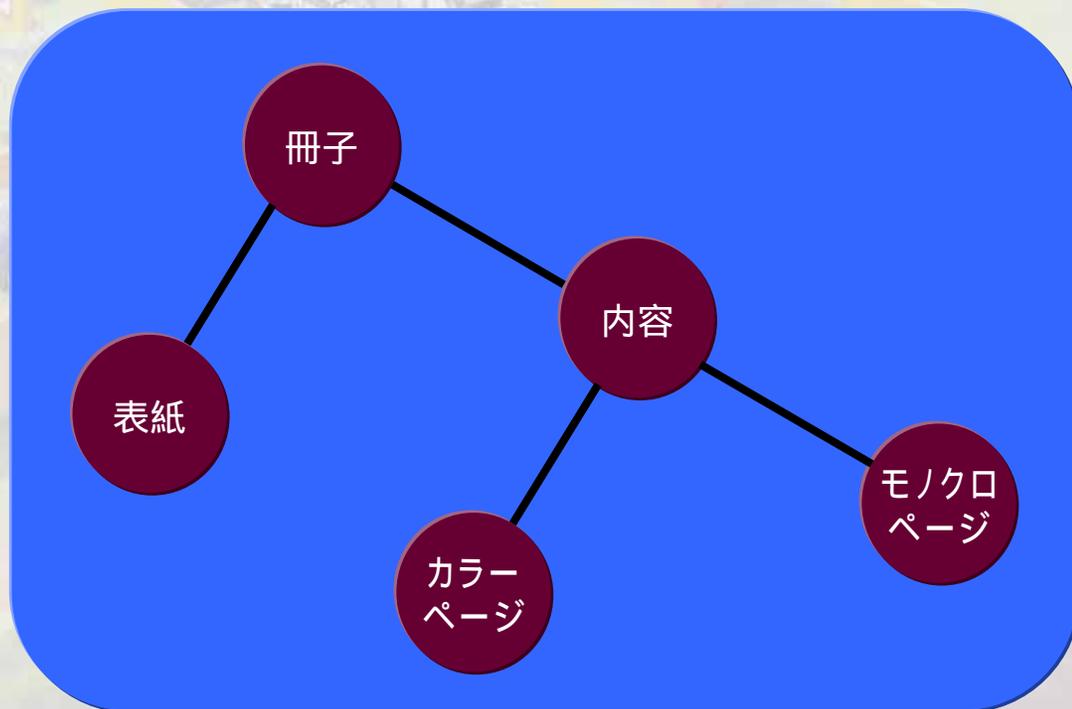




製品仕様 vs. 製造(工程)設計

- 製品仕様(プロダクトインテント)記述の目的

- 顧客が要求する「もの」に対する視点
- 契約交渉
- 工程の独立性

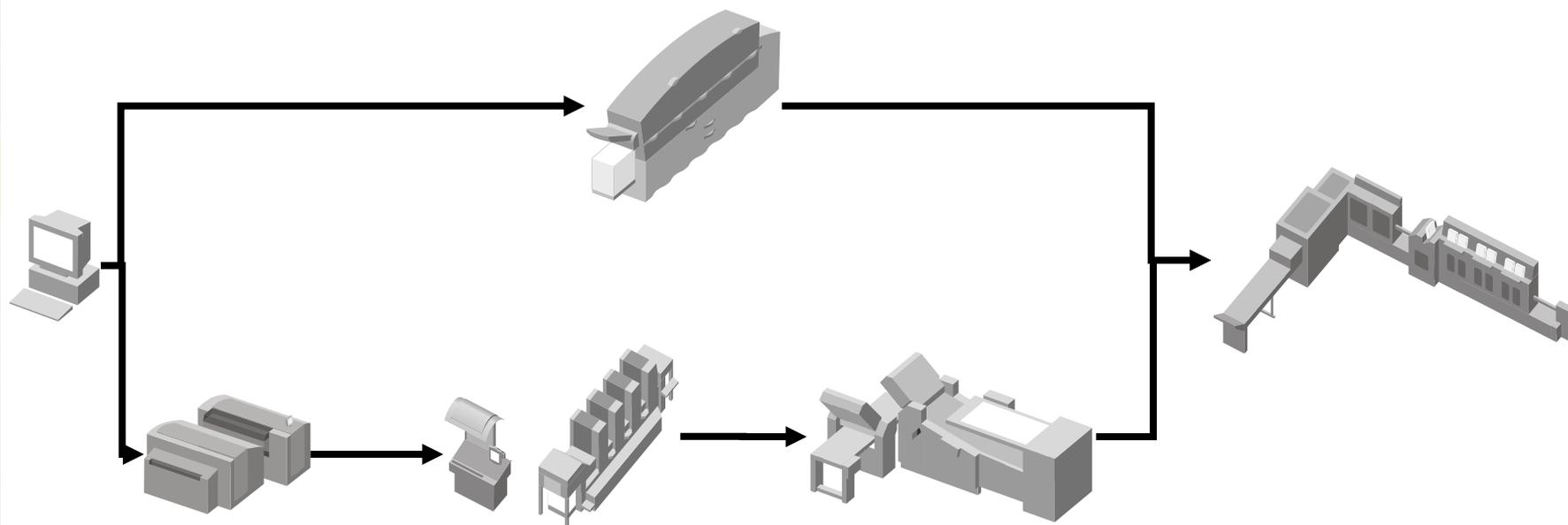




製品仕様 vs. 製造(工程)設計

- 工程モデルの目的

- MISと生産のインターフェース
- 工程連携
- 製造指示詳細





MISとグレイボックス

- MISは工程の細かな情報まではわからない
- MISから見ると
 - コストに関連するプロセスのみ重要
 - コストに関連のないパラメータは重要ではない
- 製造詳細事項をすべて把握することはできない
 - プリプレスの細かな工程群は人括りで扱われる
 - RIP、カラーマネジメント、トラッピングなどは細かすぎる
 - 面付けの概要
 - 面付け数
 - 用紙サイズ



Audit オブジェクトとは

- ジョブの実行結果を記録する
 - 実時間
 - 作業開始時間
 - 作業終了時間
 - ジョブの段階
- 仕様変更履歴を記録する
 - リソース
 - 例: 用紙の変更 (80g/m² の用紙ではなく85g/m²の用紙を使用)
 - 例: ロール紙在庫の456番ではなく123番を使用
 - 消耗品:
 - 生産量、消費量、損紙の実測値
- 状況の概要
- イベントログ



ジョブ/顧客情報

- CustomerInfo 要素

- 受注先の顧客情報を詳細に記述。あらゆるJDFノードに記述できる
- 顧客ID
- 住所(納品先、請求書送付先など)

- NodeInfo 要素

- スケジュール
- 納期
- 作業時間見積



JDFノードの一例

```
<JDF ID="N1" Type="Product" JobID="HDM1" JobPartID="p1"
  Status="Waiting" Version="1.3">
  <ResourcePool>
    <NodeInfo Class="Parameter"
      Status="Available" ID="Link0001" />
    <SomeInputResource ID="Link0002" Class="Parameter"
      Status="Available"/>
    <Component ID="Link0003" Class="Quantity"
      Status="Unavailable" DescriptiveName="SomeOutputResource"/>
  </ResourcePool>
  <ResourceLinkPool>
    <NodeInfoLink rRef="Link0001" Usage="Input" />
    <SomeInputResourceLink rRef="Link0002" Usage="Input" />
    <ComponentLink rRef="Link0003" Usage="Output" />
  </ResourceLinkPool>
  <AuditPool />
</JDF>
```





JDFノード – ノードタイプ

- 各工程の種類と各リソースの定義(例):

- インタープリティング(RIP)
- トラッピング
- 従来の印刷
- デジタル印刷
- 裁断
- 折り
- 検品
- 納品
- ...



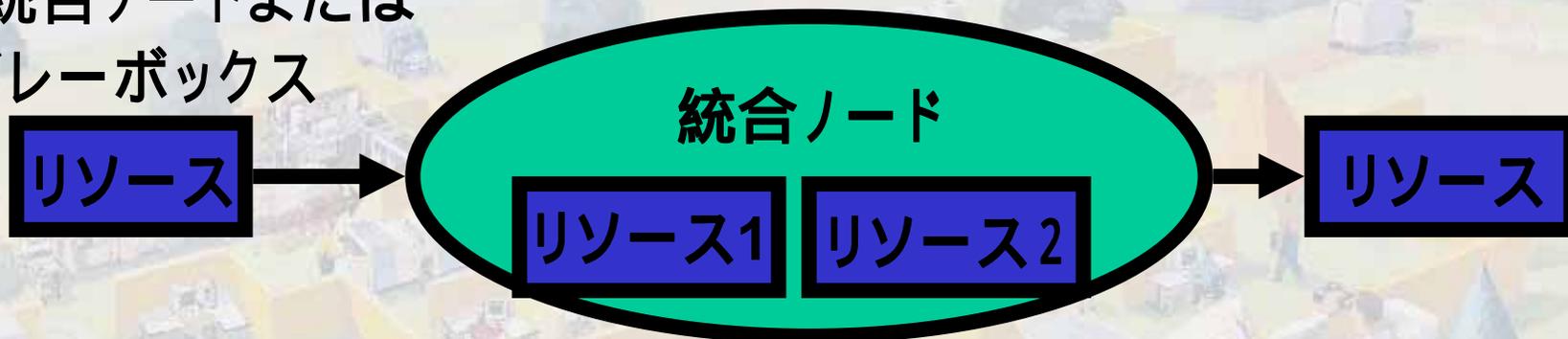
JDFノードの統合

- 複数の工程をまとめて一つの工程として扱う場合、既存の工程の代用として新しく工程の種類を定義しないこと
- 複数の定義済の工程を統合し、一つの工程にする：
 - オンライン仕上げ = 印刷 + 折り + 裁断
 - RIPトラッピング = トラッピング + リッピング
- 3種類の統合ノード
 - 統合ノード: 内部インターフェースへはアクセス不可能
 - スマートマルチファンクションデバイス
 - 工程グループ: 内部ノードへはアクセス可能
 - サブ要素がない場合 (グレイボックス)
 - MISの視点
 - サブ要素がある場合
 - 各部のワークフローグループ
 - 外注

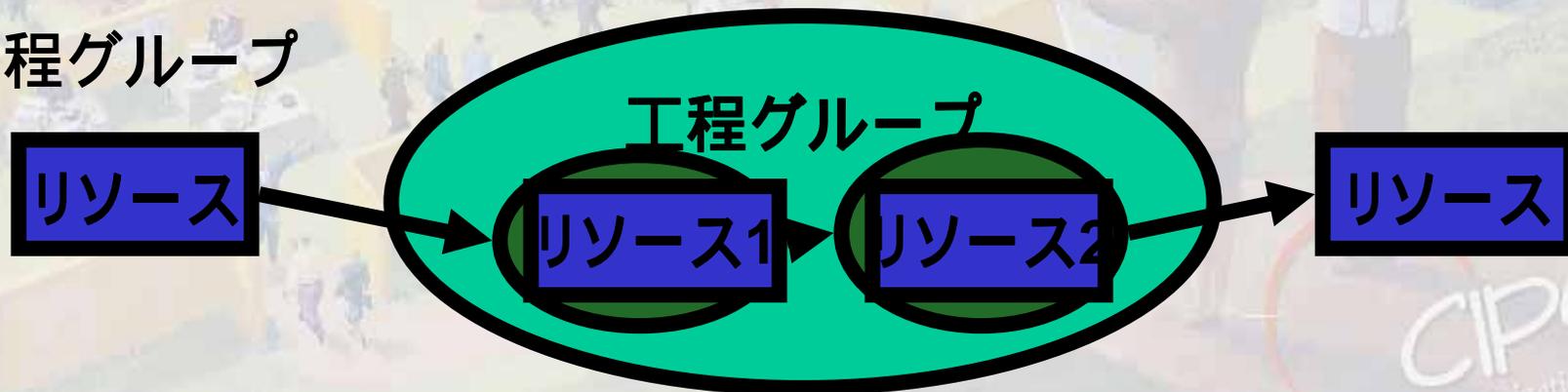


統合ノード vs. 工程グループ

- 統合ノードまたは
グレーボックス



- 工程グループ





JDFリソース

- 下記パラメータの仕様
 - 製品インテントの記述
 - 論理要素 (RIPパラメータ、面付け設定など)
 - 物理要素 (用紙、デバイス、版など)
- Adobe PJTFおよびCIP3 PPFに準拠
 - 製品インテントノードのインテントリソース
 - プリプレス: Adobe PJTF
 - プレス、仕上げ: CIP3 PPF
- JDF内部または外部に対し定義されるフォーマットへのリンク
 - サムネイル、プレビューファイル、ICCプロファイル、コンテンツデータ



JDF RunListリソースの一例

```
<RunList ID="Link0003" Class="Parameter"
  Status="Available" PartIDKeys="Run">
  <RunList Run="1" Pages="0 ~ 10">
    <LayoutElement ElementType="document">
      <FileSpec URL="File:///File1.pdf"
        MimeType="application/PDF"/>
    </LayoutElement>
  </RunList>
  <RunList Run="2" Pages="2 ~ -1">
    <LayoutElement ElementType="document">
      <FileSpec URL="File:///File2.pdf"
        MimeType="application/PDF"/>
    </LayoutElement>
  </RunList>
</RunList>
```





JDFリソース – インテントリソース

- パラメータは属性ではなく、以下のような特徴を持つ Span要素を用いて表現する:
 - レンジ: 許容値のリスト
 - 名前
 - 数
 - 数/ストリングの範囲
 - オフアールレンジ: 予め指定された値のリスト
 - プリファード: 優先値
 - アクチュアル: レンジまたはプリファードから選択された値



JDFリソース - 詳細

- 複数パートの特別処理:
 - シート(折丁)
 - 分版
 - 損紙
 - タイル
 - バージョン
- リソース内の一部だけを再実行
 - シート17番の表面の黄版のみ
- 分割リソースの並行処理
- デバイス/オペレータの選択



JDFリソース – 分割

- 1つのリソースが複数の要素を決定する場合もある
- 複数の分割タイプレベル(シート、サイド(表裏)、分版など)
- 相続モデル
 - 共通属性の定義は一回のみ
 - 各属性/要素を上書き
- 各パートまたは複数パートをリソースリンクで選択

分割リソース共通パラメータ

シアン色分解パラメータ

マゼンダ色分解パラメータ

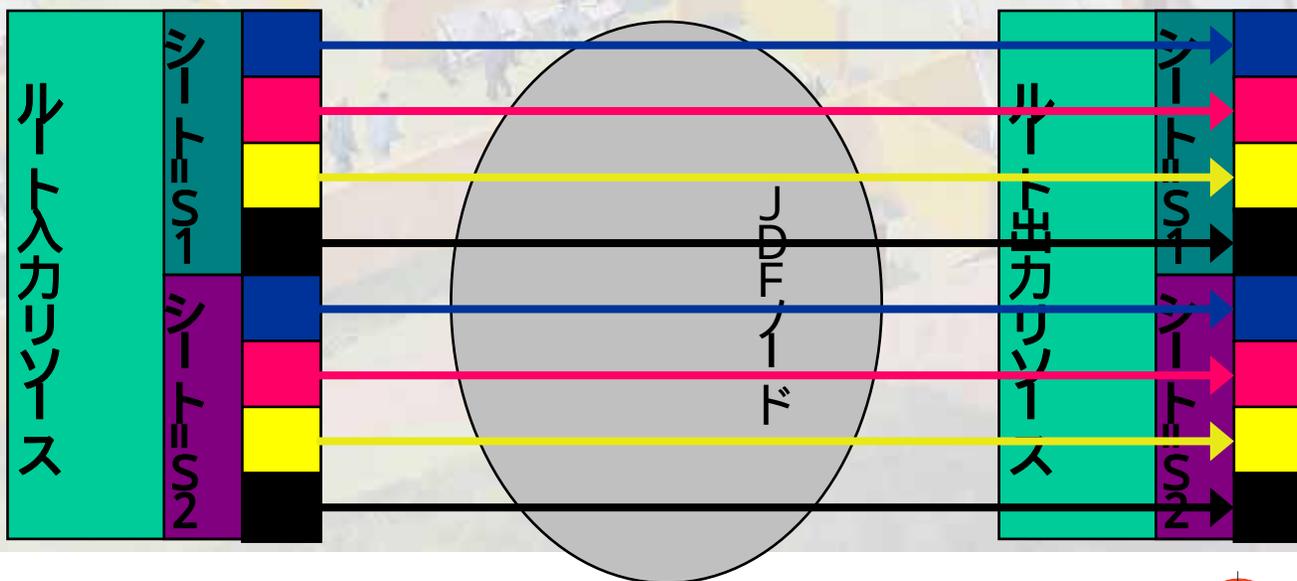
イエロー色分解パラメータ



分割とノード

分割リソース

適切な分割の重要事項が関連している



CIP4[™]
Organization



分割インクリソースの例

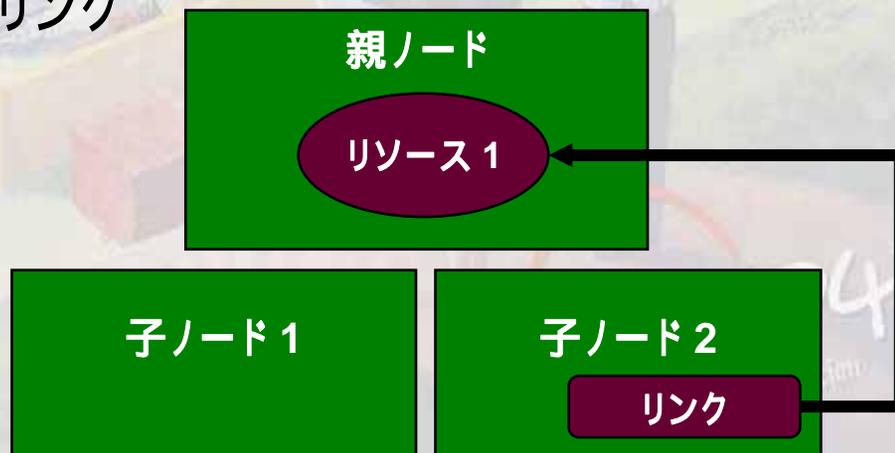
```
<Ink ID="InkID" Brand="ProcessBrand" Class="Consumable"
  Status="Available" MediaType="Coated"
  PartIDKeys="Separation">
  <Ink Separation="Cyan">
    <Color CMYK="1 0 0 0"/>
  </Ink>
  <Ink Separation="Magenta">
    <Color CMYK="0 1 0 0"/>
  </Ink>
  <Ink Separation="Yellow">
    <Color CMYK="0 0 1 0"/>
  </Ink>
  <Ink Separation="Black">
    <Color CMYK="0 0 0 1"/>
  </Ink>
  <Ink Brand="SpotBrand" Separation="Heidelberg Spot Blau">
    <Color CMYK="0.7 0.7 0.3 0.7" ColorantUsage="spot"/>
  </Ink>
</Ink>
```





リソースリンク

- リソースとノードを関連づける
- リソースの使用方法を定義（入力または出力）
- リソースのサブセット/パートへのリンク
 - 例えば、シート1番の表紙をシアン版にする
 - 量の決定
- 複数の工程によるリソースの再使用が可能
 - 1つのリソースが複数のリソースリンクの中で指定される場合もある
- リンクにより工程連携を決定





リソースリファレンス

- 別のJDF要素内のリソースをリファレンスとして使用
- 個々の区分を選択するための部分的な要素を含む場合もある
- リファレンスリソースのインライン実現値に相当
- 名前付けルール: リソース名 + 「Ref」



リソースリファレンスによるJDFランリスト リソースの一例

```
<RunList ID="Link0003" Class="Parameter"
  Status="Available" PartIDKeys="Run">
  <RunList Run="1" Pages="0 ~ 10">
    <LayoutElementRef rRef="L1">
  </RunList>
  <RunList Run="2" Pages="12 ~ -1">
    <LayoutElement ElementType="document">
      <FileSpec URL="File:///File2.pdf" MimeType="application/PDF"/>
    </LayoutElement>
  </RunList>
</RunList>
<LayoutElement ID="L1" ElementType="document" Class="Parameter"
  Status="Available">
  <FileSpec URL="File:///File1.pdf" MimeType="application/PDF"/>
</LayoutElement>
```

リソース
リファレンス

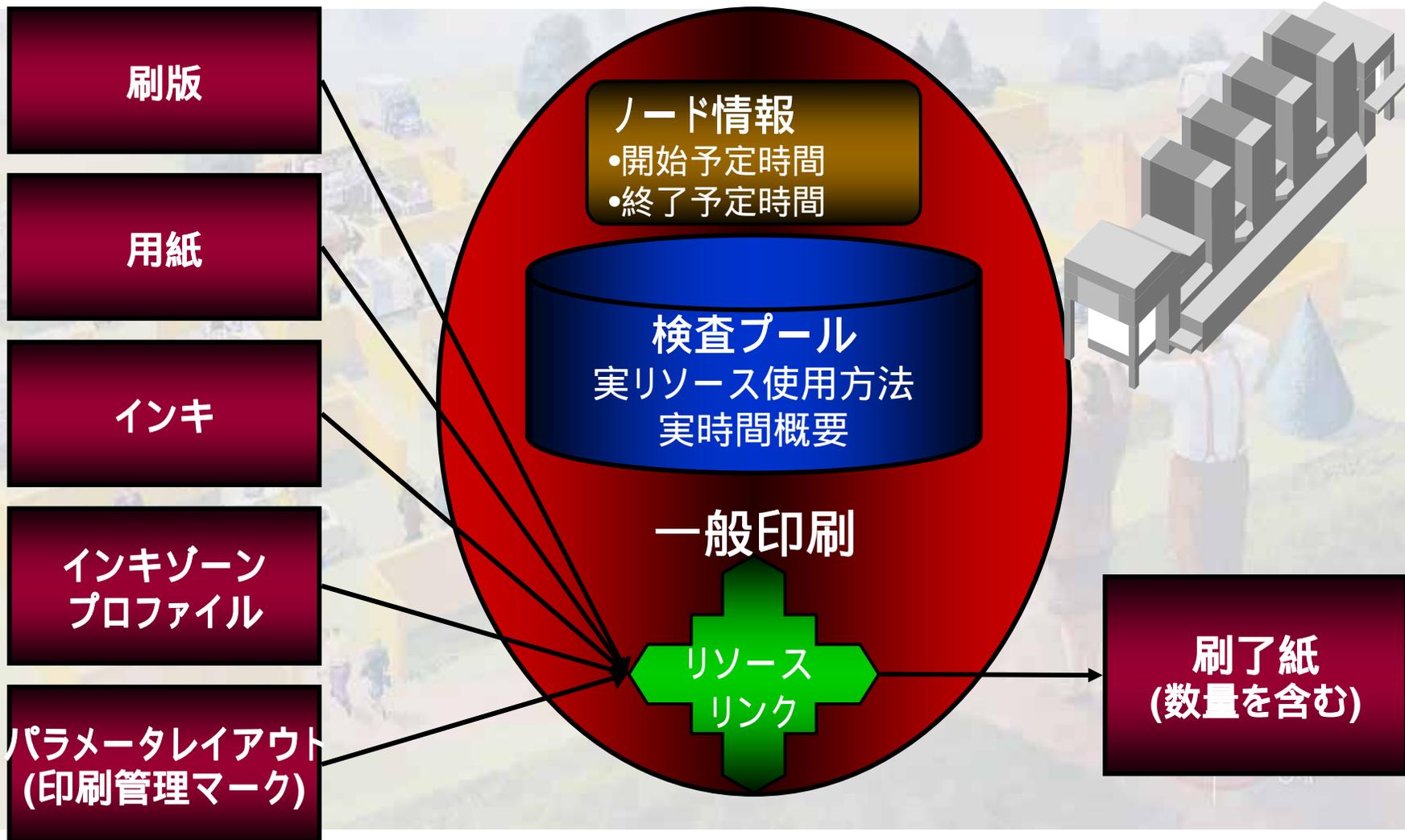
インライン
リソース

リファレンス
リソース





一般印刷のJDFノードの例





JDFノード – 実行のための要求事項

- 要求された入力リソースがすべて有効であればノードは実行可能
 - 連続処理 – 標準
 - 並行処理 – 部分的リソース
 - オーバーラップ処理 – パイプ
 - 反復処理 – ドラフトリソース
- ノードを相互依存させることで工程を配置できる
 - 印刷ノードの作業に必要な顧客承認を校正ノードが作成



ノードの実行可能性

リソース

リソースリンク

JDFノード

無効

有効

有効

実行不可能

リソース
リンク



ノードの実行可能性

リソース

リソースリンク

JDFノード

有効

有効

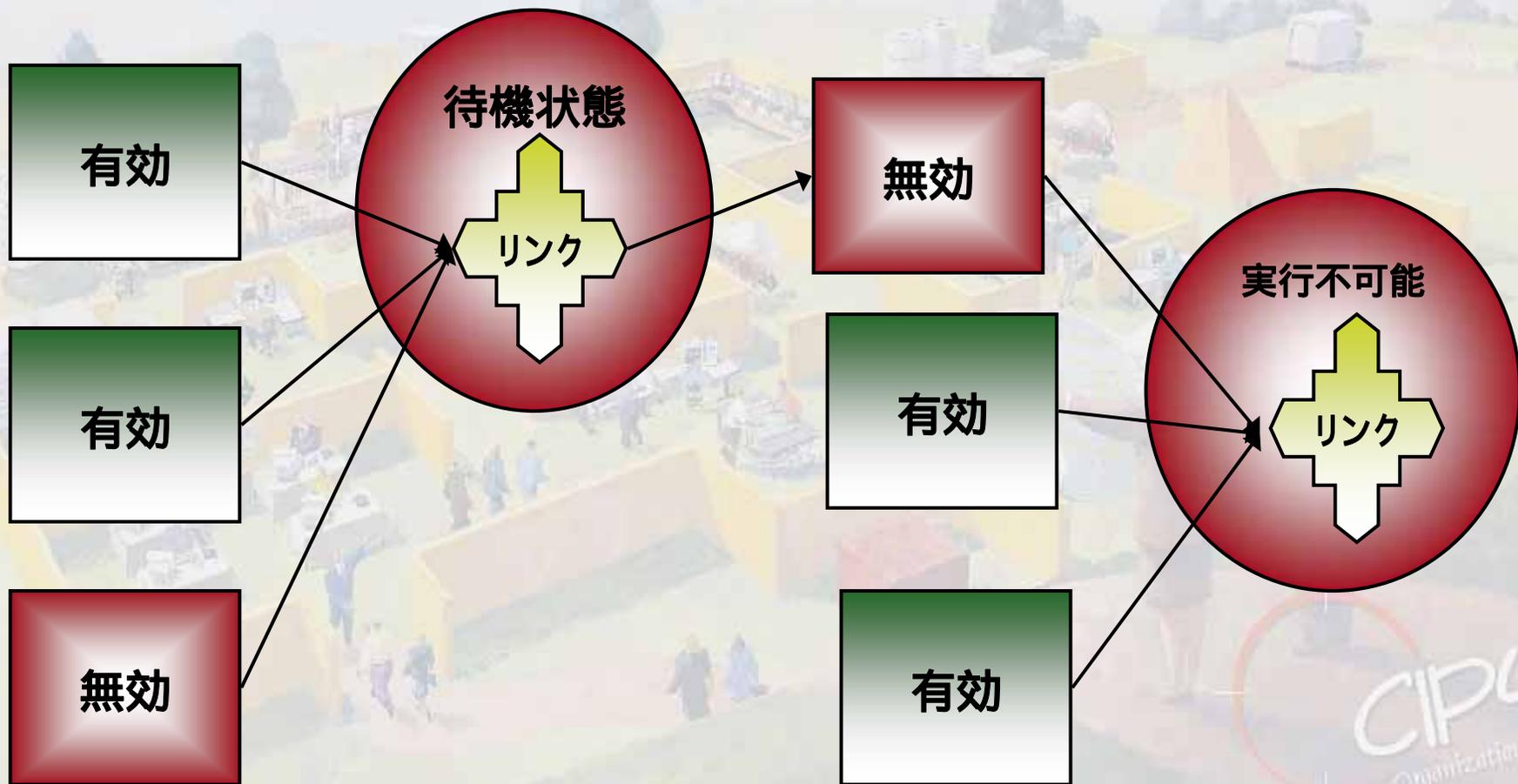
有効

実行可能

リソース
リンク

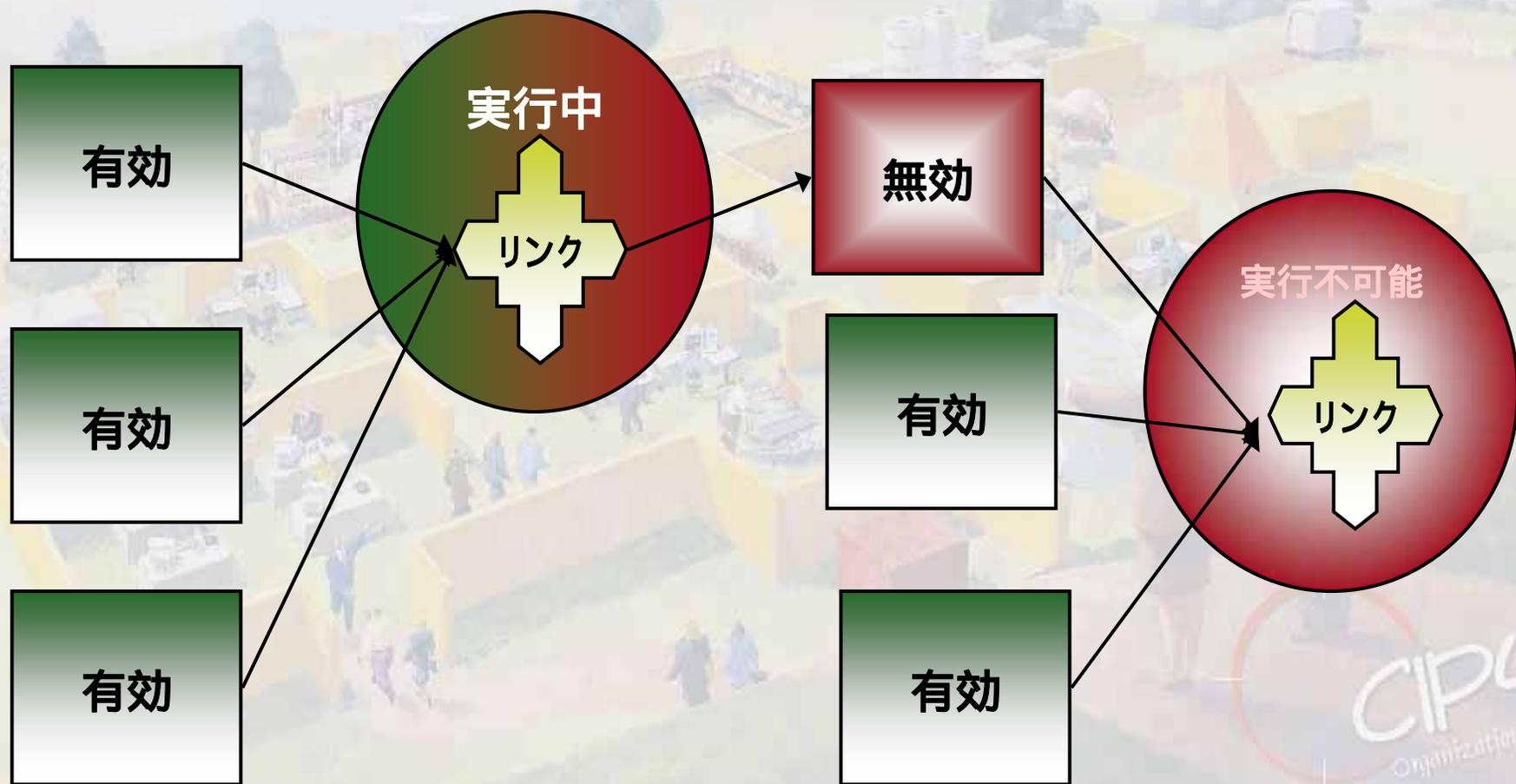


ノードの実行可能性とネットワーク



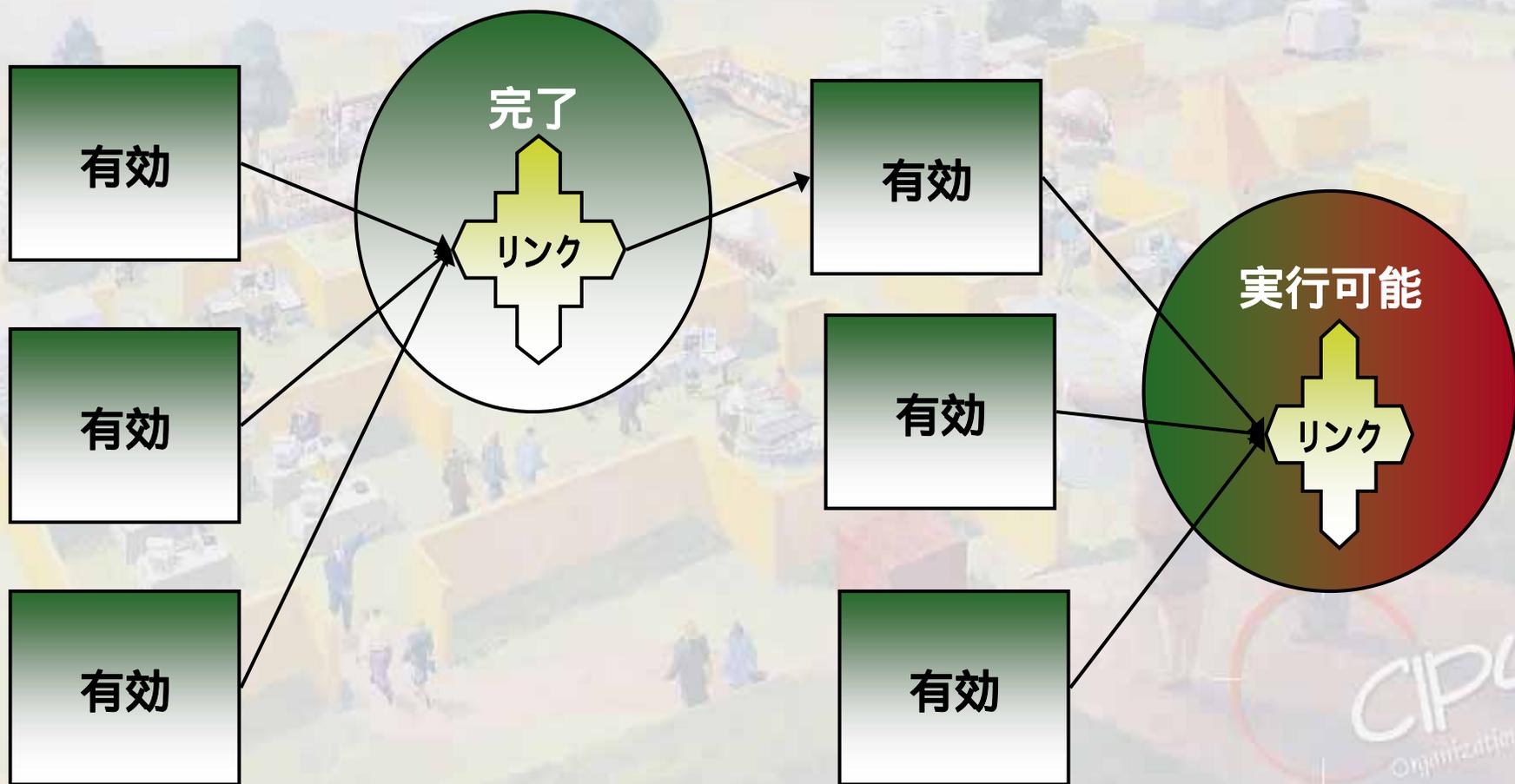


ノードの実行可能性とネットワーク





ノードの実行可能性とネットワーク



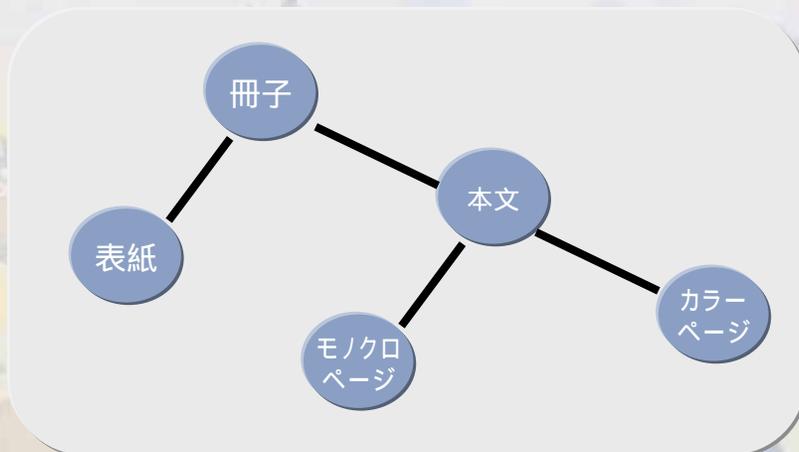


JDF I がサポートするジョブ記述モデル

- 製品仕様定義

- 工程情報なし
- 顧客の視点
- 製品コンポーネント別区分

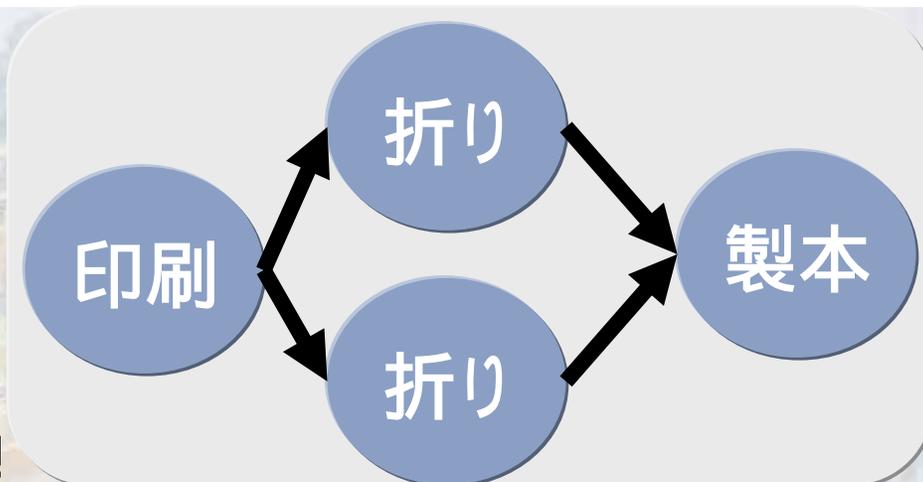
- 連続処理





JDF II がサポートするジョブ記述モデル

- 並行処理



- オーバーラップ処理

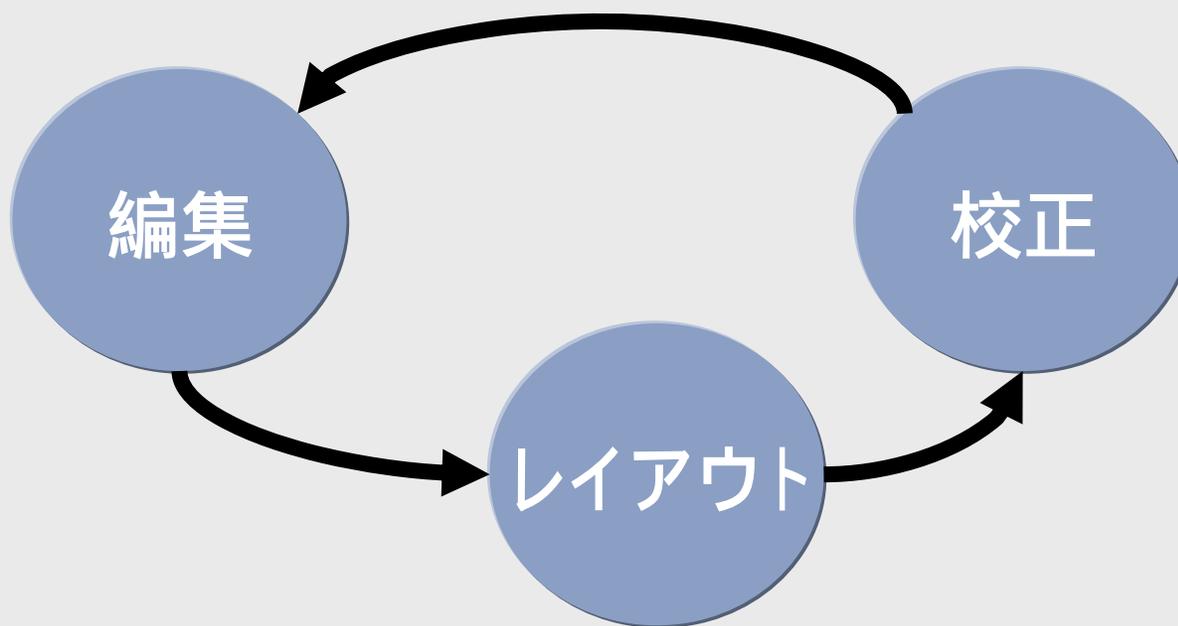




JDF III がサポートするジョブ記述モデル

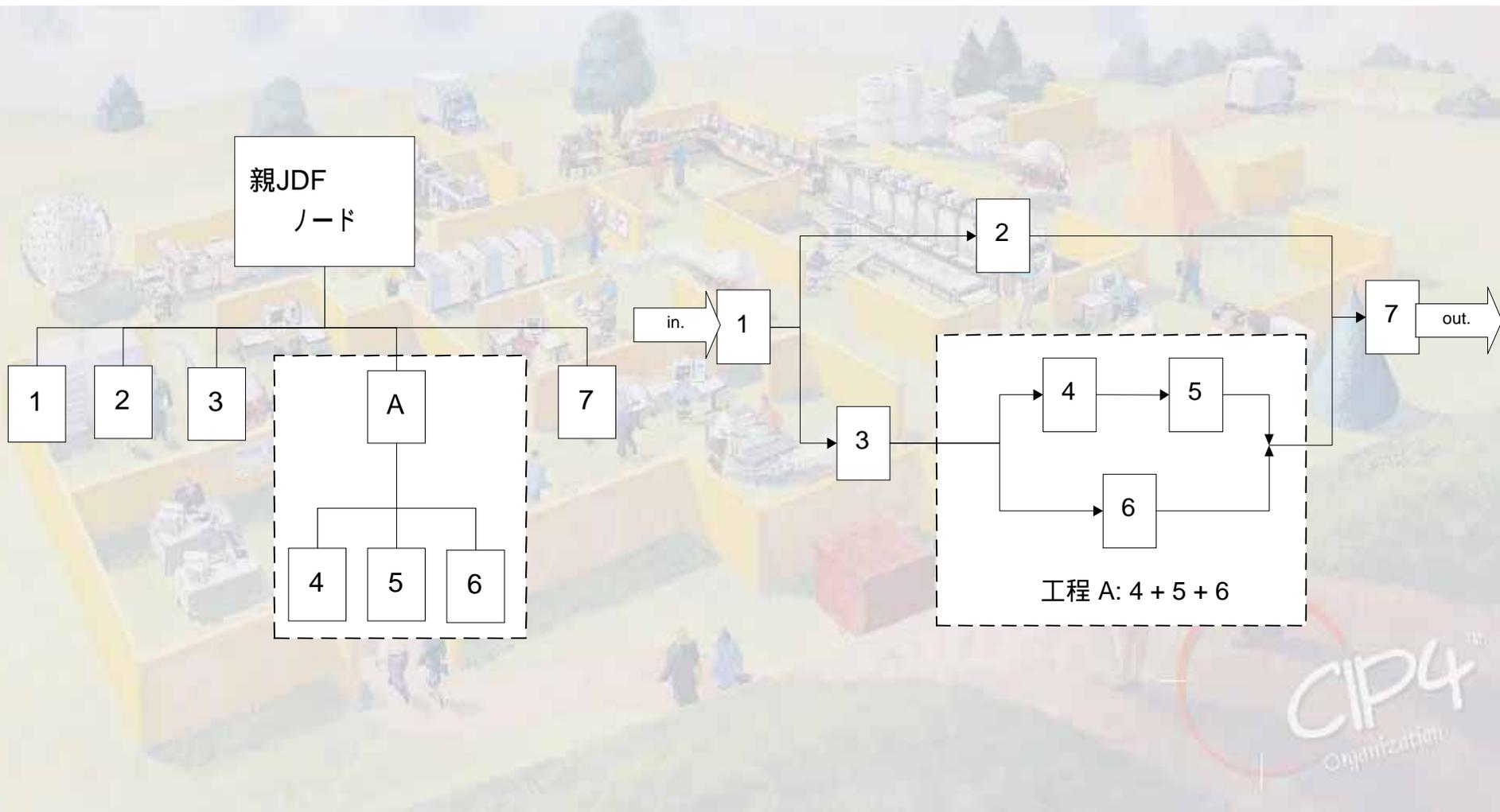
- 反復処理

- ドラフトリソースによる非定型反復処理
- JMFメッセージによる定型反復処理





JDFツリー/ネットワーク構造



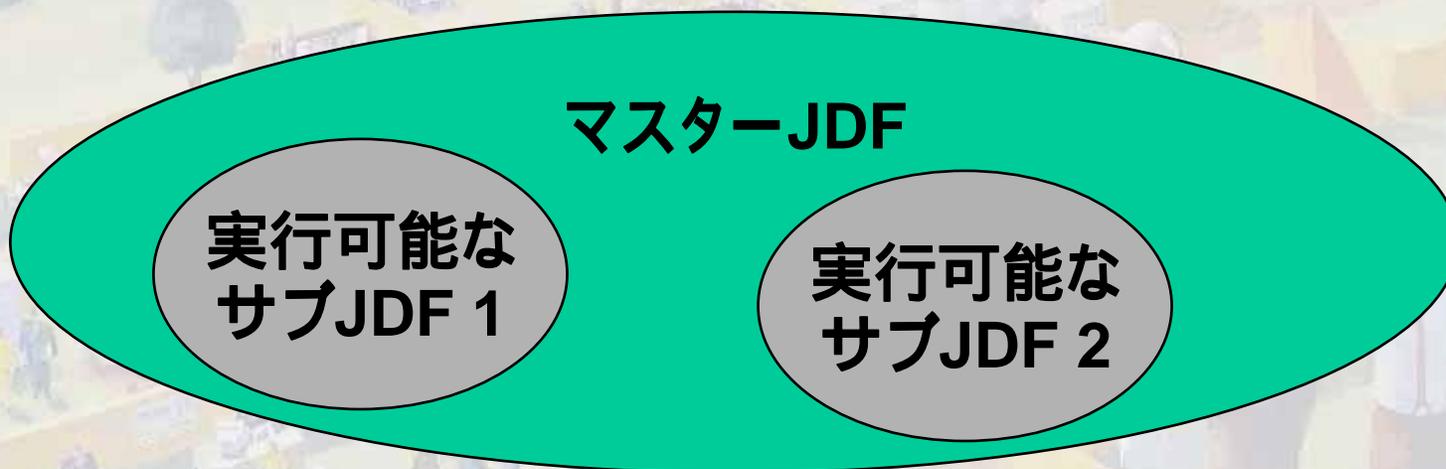


JDF – 分割と統合

- 独立した処理を行うためにJDFツリーの各ノードを分割
 - 並行処理
 - 外注
 - 分割処理をサポート (例: シート1番のみ)
- 処理後に統合
 - 情報源のコンテキストを保持
 - 検査
 - 変更リソース



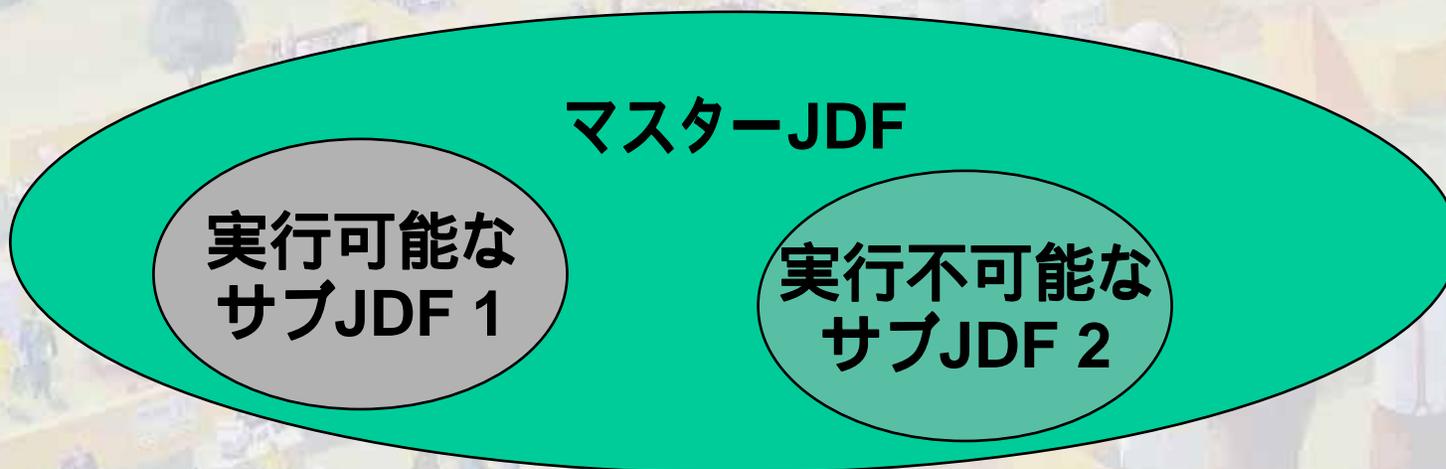
分割と統合



ステップ1 -
分割前



分割と統合



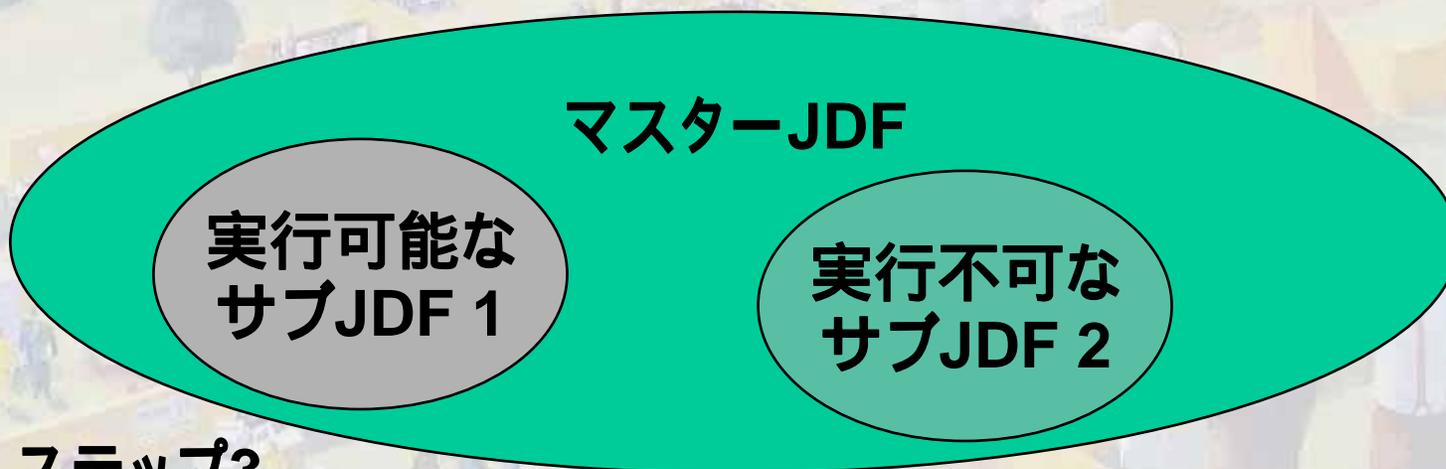
ステップ2 -

分割後、実行前





分割と統合



ステップ3 -

分割後、

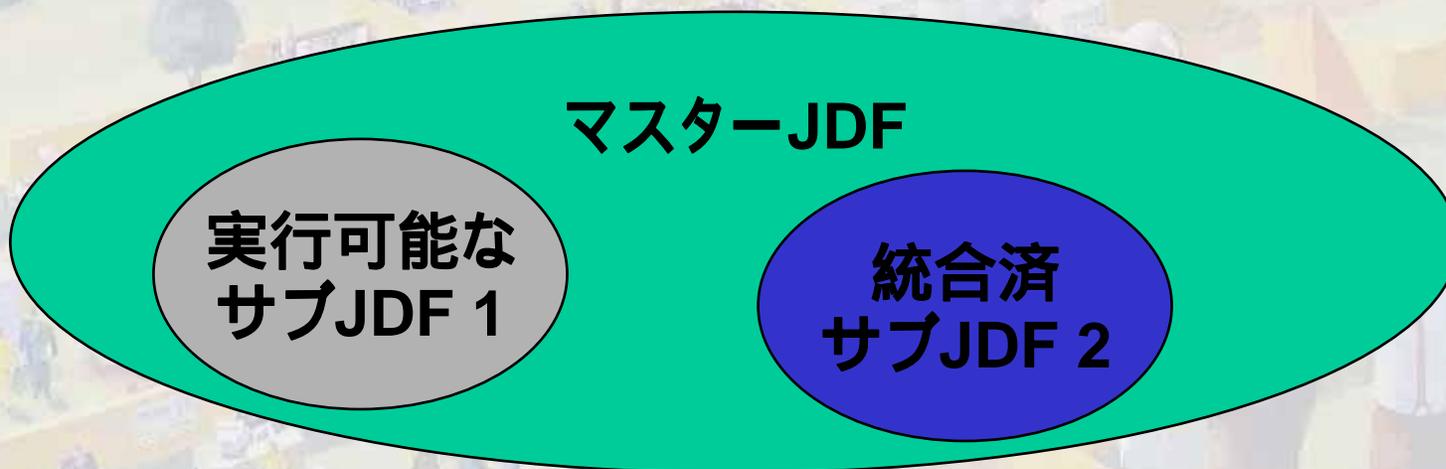
単独実行後、

統合前

分割後・実行済
サブJDF



分割と統合



ステップ4 -
実行後、
統合後



JMFメッセージ

- リアルタイムデータ交換フォーマット
- 小さなXML構造
 - JDFはURLを介して参照できるが、メッセージとは結合していない
 - 構想：ネットワーク通信量を最小にし、実際に必要となるまでデータの送信を遅らせる
 - 有用であると思われる情報が利用可能になれば、すぐに送信する



JMFメッセージ

- 使用目的:
 - ジョブ/デバイス状態のスナップショット
 - 動的ジョブの更新
 - ジョブの提示と待機/待機入力の処理
 - 機能の発見
 - (将来的に)プラグ&プレイ・ブートストラッピング
- 一般的にイントラネット内で使用される



JMFメッセージグループ

- 6種類のJMFメッセージ

- 指令 (command)

- レシーバは実行あるいは変更の指示を受ける

- 照会 (query)

- レシーバは情報の返信指示を受ける。実行や変更は行わない

- 応答 (response)

- 指令や照会の結果を即時返信。
- メッセージの同時送信 – 指令/照会や応答を同時にオープン接続で交換



JMFメッセージグループ

• JMFメッセージの分類(続き)

– 確認 (acknowledge)

- 指令からの指示後、しばらくしてコマンド結果を返信する
- メッセージ非同期送信 – コマンドの指令を受けた同じ接続で空の応答を即時返信。確認応答を後に送信する。

– 信号 (signal)

- イベント通知の送信やステータスの変更を行う
- 一般に信号内のサブスクリプションによる照会結果
- サブスクリプションが継続的な経路を設定

– 登録 (registration)

- 指令を要求する
- 三角関係となるワークフローの設定に使用される
- (例) MISがプリプレスに、プレスへの指令を送信するよう指示を出す



SubmitQueueEntryメッセージの例

```
<JMF DeviceID="SP013" SenderID="MIS1"
  TimeStamp="2006-05-05T12:32:48-06:00">
  <Command ID="m3829"
    Type="SubmitQueueEntry">
    <QueueSubmissionParams
      URL="http://jobserver/getJob?job=10047"
      ReturnJMF="http://jobserver/jmfservice"
      Priority="50"/>
    </Command>
  </JMF>
```





JMFメッセージの種類

- JMFメッセージの例は6種類
 - 指令 (command)
 - SubmitQueueEntry, AbortQueueEntry, ReturnQueueEntry, HoldQueue, ResumeQueue
 - 照会 (query)
 - KnownControllers, KnownDevices, Status
 - 応答 (response)
 - 多様な指令/照会に関連
 - 確認 (acknowledge)
 - 多様な指令/照会に関連
 - 信号 (signal)
 - Status, Resource, Occupation
 - 登録 (registration)
 - Resource



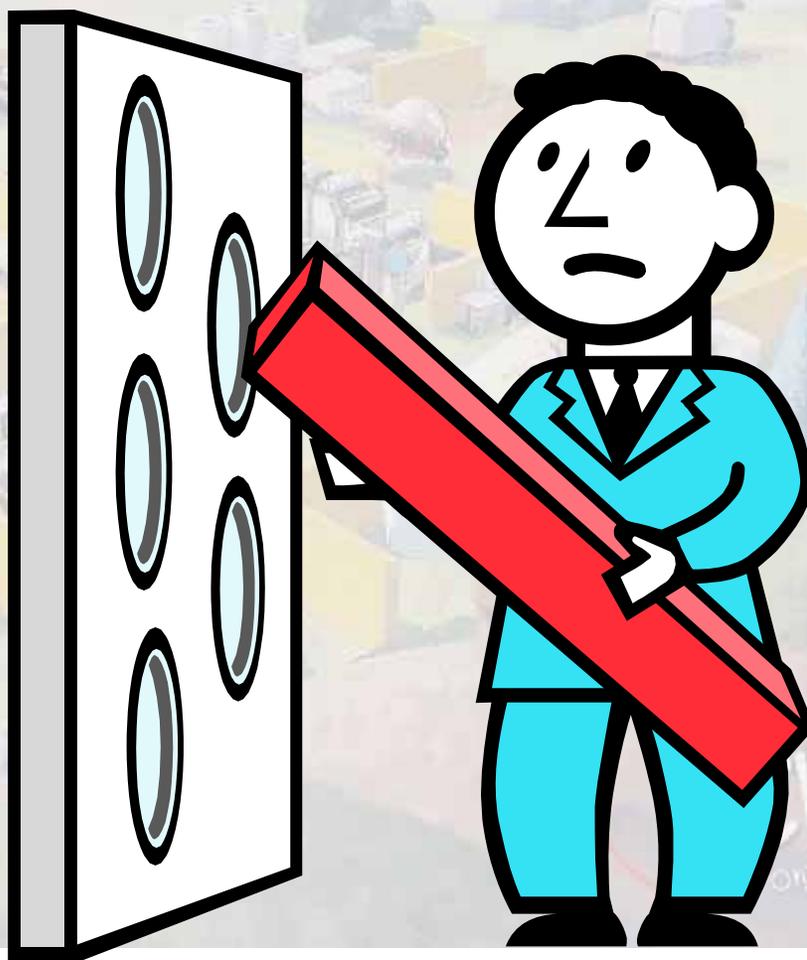
今後のJDF機能

- 相互運用性
- 機能
- ワークフロー
- PPDとスキーマ
- 機能理論
- 機能概要



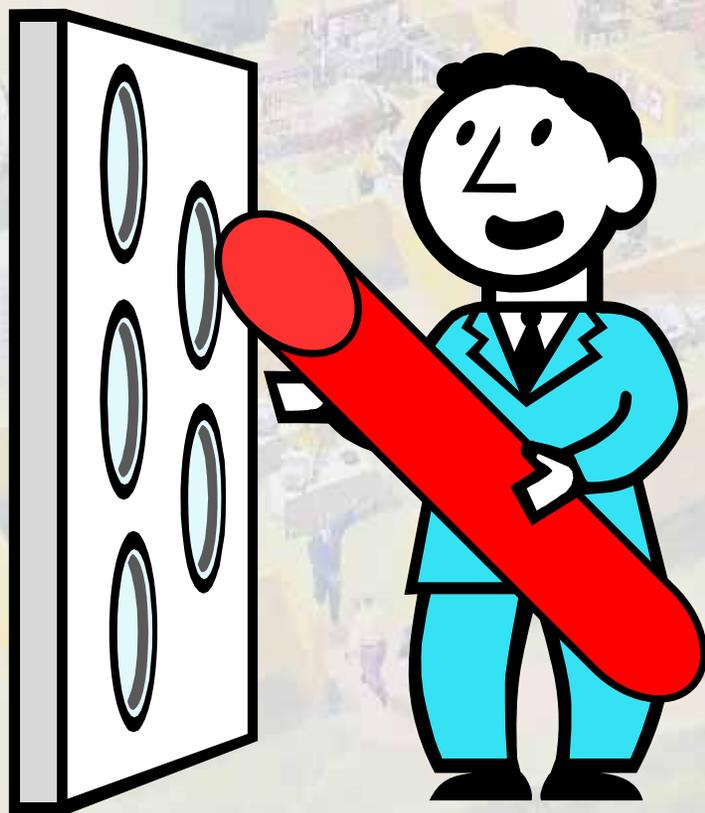
相互運用性とは？

機能を知らなければ
「相互運用」は不可能





相互運用性とは？

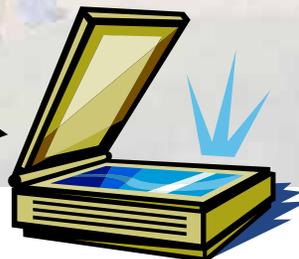


しかし、機能を知っていれば
正しいツールを使うことが可能



「機能」とは？

「機能」によりワークフロー行為者は
他の行為者が実行できることを決定できる





機能の使用目的

- 印刷会社の生産する製品を決定
 - JDFIntentの作成が可能（製品記述）
- 印刷会社が利用できる装置を決定
 - MISシステムはJDFIntentから生産方法を決定し、ジョブを作成する
- ユーザ・インターフェースを作成し、各装置やアプリケーションのジョブの処理を構築する
- JDF消費者が処理をする前にJDFをプリフライトする



PPDおよびスキーマとは？

PPD(ポストスクリプト・プリンタ・ドライバ)

- プリンタに関する基本的な機能情報を提供する
- 一部のポストスクリプトを特定することによって、プリンタオプションの設定が可能
- ユーザ・インターフェースの制約メカニズムを提供する
- コントロールをローカライズする
(ただし、一般的に「詳細」ダイアログに対してのみ)



PPDおよびスキーマとは？

スキーマ

- JDFの全体的な機能を記述する（工程とリソース）
- 制約の定義はサポートしない
- ローカライゼーションはサポートしない
- デバイスに全体的な機能のサブセットを特定させない



では機能とは？

- XMLベース
 - 当然のことながら
 - 機能の記述はJDF構造を直接反映している
- スキーマに対するデバイスの導入を制約
 - JDFスキーマまたは拡張スキーマは、
 - 作業で使用するスキーマを必要としない

注：
スキーマはJDFの
全体的な機能を特
定するのみ





しかし、それだけではありません





機能理論

JDFは下記に基づく機能モデルを定義する

- デバイスとはJDFノードを実行する「もの」である
- JDFノードは「リソース+リソースリンク」により定義される
 - 例: プレスは各々の許容媒体とプレスパラメータを有する



機能理論

JDFパラメータとは

- 各々が独立している。または
- 独立した許容装置の一覧として表現される。
(例: 1000dpiグレイスケールまたは500dpiカラーでスキャンできるスキャナ)
- 論理(ブール)式によりさらなる制約が可能
 - デバイスで両面印刷が可能
 - デバイスでスライド印刷が可能
 - 両面印刷とスライド印刷の同時実行は制限される

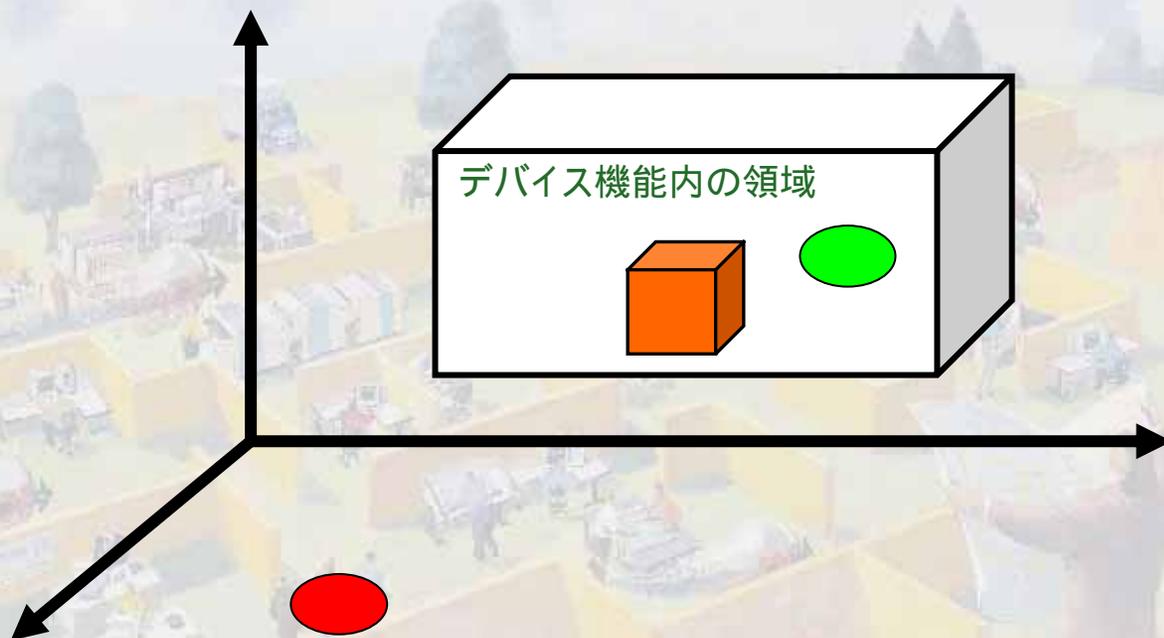


機能理論

- 性能を特定できる
 - 最大値 + 平均値
 - ジョブパラメータに依存
- 現行の永続的な機能を特定できる
- デフォルトおよびサポートする特徴/サポートしていない特徴を定義する
- 異質環境におけるプラグ & プレイに対するJMFメッセージ
- ローカライゼーションを特定する



デバイスパラメータ空間



-  有効パラメータポイント
-  無効パラメータポイント
-  制約領域



立体に対する制限は表示目的のみ





機能概要

特徴のマクロ視点 (JDF 1.2)

- 複数のJDFコントローラをユーザ・インターフェースの1つの抽象コントロールで表現する
 - スキャナ・クオリティ: (draft, good, best)
 - 解像度、カラーモードなど



機能概要

特徴の利用可能性 (JDF 1.2)

- デバイスにインストールされており、利用可能
- デバイスにインストールされていない
- デバイスにインストールされているが、許可されるまで利用不可能
- デバイスにインストールされ、許可されているが利用不可能



機能概要

ユーザ・インターフェースとは (JDF 1.2)

- 特徴のグループ
- 単位の種類
 - 角度、長さ、重さなど
- 特徴を表示すべきか
 - 表示すべき
 - 表示すべきではない
 - 管理者にのみ表示すべき
 - 条件付きで表示すべき
 - ベンダーに有用な場合のみ表示すべき
- デバイスを表すアイコン



PrintTalk – www.printtalk.org

- JDFは生産予定の製品を記述する
- PrintTalkは製品の業務状況を記述する
 - 見積要求
 - 見積
 - 発注
 - インボイス
 - 変更依頼
- PrintTalkは価格情報の管理も行う





CIP4 オープンソース開発ツール

- XMLスキーマ
- C++ API
- Java API
- JDF Editor
 - JDF + JMFの視覚化
 - メッセージの送信
 - JDF + JMFの更新・検証
- Elkプロジェクト
 - リファレンスデバイス
- Alcesプロジェクト
 - リファレンスマネージャー
- CheckJDF – JDF検証ウェブサービス
- FixJDF – JDF更新ウェブサービス





JDFの沿革

1999: アドビ社、アグファ社、マンローランド社、ハイデルベルグ社がGATを設立。グラフィックアート業界のジョブチケットの定義およびMISの統合を目指す。

2000: CIP3が規格の所有権を獲得

2001: JDF 1.0を発行

2002: JDF 1.1を発行

- 導入時の不明瞭な点や誤りを削除
- 新工程を追加
- 矛盾箇所を訂正し、一貫性を追及

2004: JDF 1.2を発行

- 新工程を追加
- さらに内容を深め、一貫性を追求

2005: ICS仕様書の初版を発行

- JDF 1.3を発行

2006: JDF 1.3 が中心となる

- 初めての認証製品が出る(レイアウト制作/面付け)

2007: JDF 1.3 ICSリリース、JDF 1.4は認証製品が中心となる





JDFの現状 – 長所

- ☺ グラフィックアート業界においてジョブデータを特定する際の共通言語となる
- ☺ 生産、顧客、MISを統合する
- ☺ 生産、顧客、MISに対して複数の視点を持つことができる
- ☺ 生産ネットワークを定義する
- ☺ 工程を統合する
- ☺ データ構造だけではなく、伝達プロトコルを定義する
- ☺ モデルを拡張する
- ☺ グラフィックアート業界の他の規格と相互に作用しあう

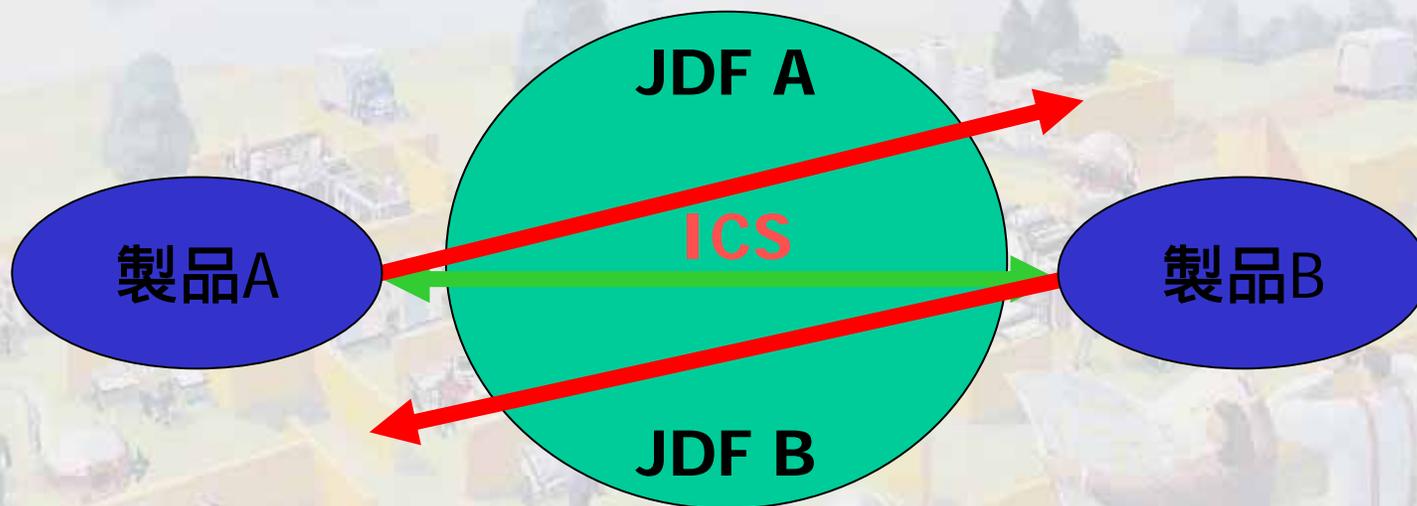


JDF現状 – 課題/問題点

- ☺ グラフィックアート業界の工程数
 - ☹ ジョブの発生
 - ☺ プリプレス
 - ☺ プレス
 - ☺ 仕上げ
 - ☹ 発注処理
- ☹ 規格の複雑さ
 - ☹ 仕様のサイズ
 - ☹ 抽象的な構想を具体的に実現する
 - ☹ 仕様の適応性
 - ☹ デバイスとのインターフェースにおいて生産ネットワークの役割は明確ではない
 - ☹ 個々の拡張
 - ☹ 抽象的なコーディネートシステムの定義
- ☺ 従来のシステムの寿命



なぜICS ?



相互運用性
適合仕様書

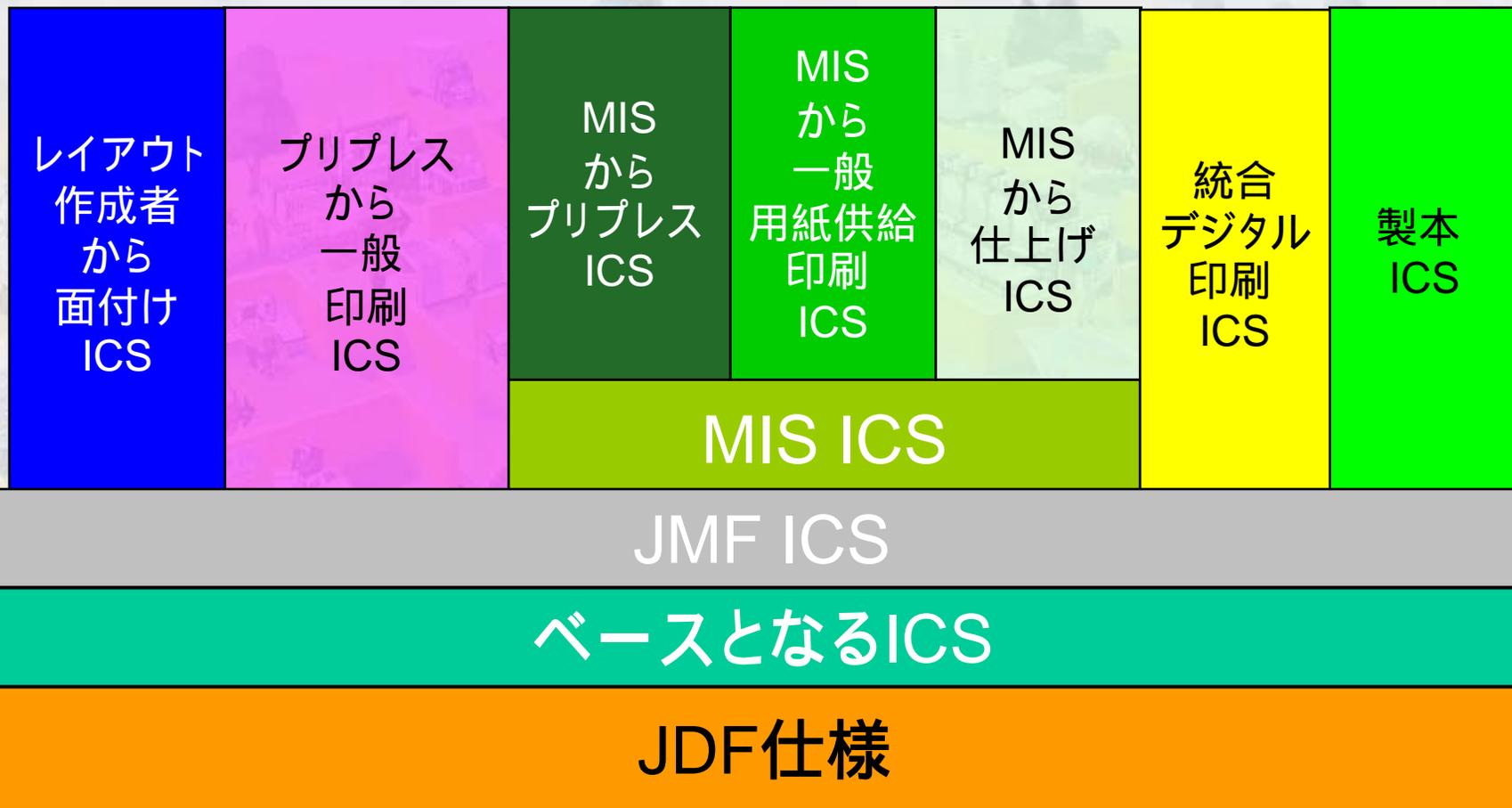


相互運用性適合仕様

- 相互運用性適合仕様 (Interoperability Conformance Specifications) の定義 – ICS
 - 「JDF適合」とは、任意の2つのアプリケーションが情報を交換していることを直接に示すものではない
 - RIP指示を折り機に送信することは有用ではない
 - 任意のICSに適合している2つのアプリケーションが有用な方法で情報交換する
 - 製品認証はICSコンプライアンスをベースとしている
- 互換性がなく重複する**JDF専門用語は使用しない**



ICS仕様書の構成





ICS仕様書

- 認証のための基準を提供する
- 製品が特定条件を必ず読み/書き/受領するようにする
 - 各アプリケーションの相互運用性
- 初版のICS仕様書は機能が制限されていた
 - 追加機能は選択できるが確実ではない
 - レベルが複数に及ぶ
 - 今後のICS仕様書には機能をさらに追加する予定



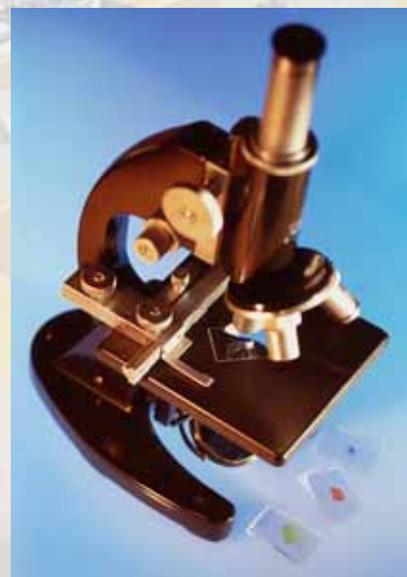
読み – 書きの適合性

- **JDF仕様**: パラメータの有効基数をJDFインスタンスやJMFインスタンスに記述する
- **ICS**: アプリケーションやデバイスの適合必要条件を特定
 - **W**: 書き込み必須。常時パラメータを書き込むと、アプリケーションだけが適合する。
 - **W?**: 書き込み可。適合の必要条件ではない。
 - **W ←**: 状況に応じて書き込む。各アプリケーションを必要とする外部条件が存在する場合。
 - **!W**: 書き込み禁止。パラメータを書き込むと、アプリケーションが適合しない。
 - **R**: 読み込み必須。情報は「**構文解析し、適切に処理するべき**」
 - **R←**: 状況に応じて読む。サポートを必要とする条件が存在する場合
 - **R?**: 読み込み可。適合の必要条件ではない。



CIP4の製品認証

- 特定のICSを認証
- 製品のJDFインターフェースの性能詳細
- 認証済アプリケーションの相互運用性を保証
- CIP4の代理としてPIA/GATFが認証を担当





JDF 1.3: 追加事項

- 変更指示
- ロット追跡
- 進捗状況メッセージ
- マニュアル作業の追跡
- 梱包 & ラベリング:
 - 梱包用接着剤
 - 箱組み立て機
 - バーコード
 - 緩衝材
 - ステップ&リピート面付け
- ウェブ & 新聞
 - 輪転機の配置
 - ウェブパス
- デジタルプリント
 - クリックカウンタ
- プリプレス
 - フォーマット変換
 - 外部面付けテンプレート
 - バージョンサポート
 - ページ要素の追跡



JDF 1.4: 概略

- 「最終指示」2008年秋
(互換性テスト)
 - フォスターシティ
- 最終版:
 - 2007年冬/2008年
- 発行:
 - 2007年春
- プレスリリース: Drupa
- コンテンツの作成
- 動的マーク
- 改善された自動面付け
- 梱包とラベリング
- JMFセキュリティ
 - 信頼性
 - 真正性
- 大貼り用のジョブ
- 変更指示
- ...



JDFチュートリアル

ご静聴ありがとうございました

