FUJITSU Software INTARFRM ご紹介

２０２６年２月

富士通株式会社

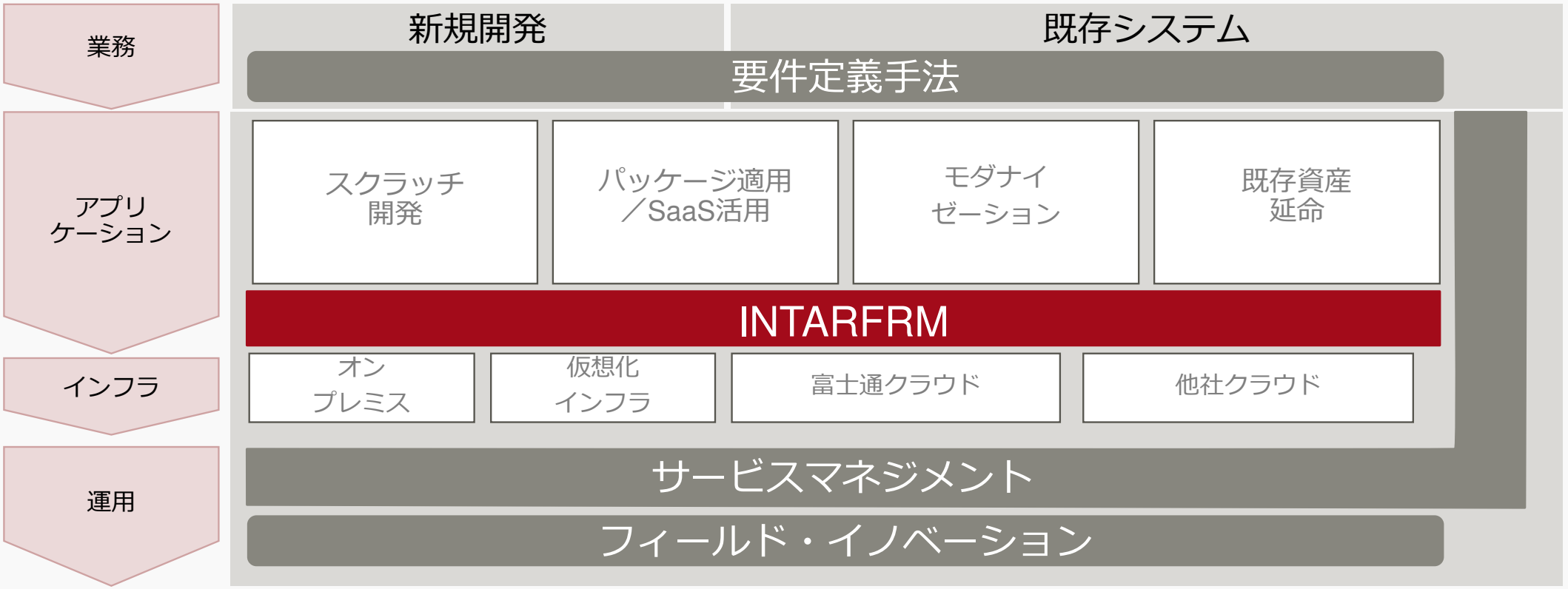
目次

FUJITSU

1. INTARFRMの位置付け

システムのLCMを支える INTARFRM

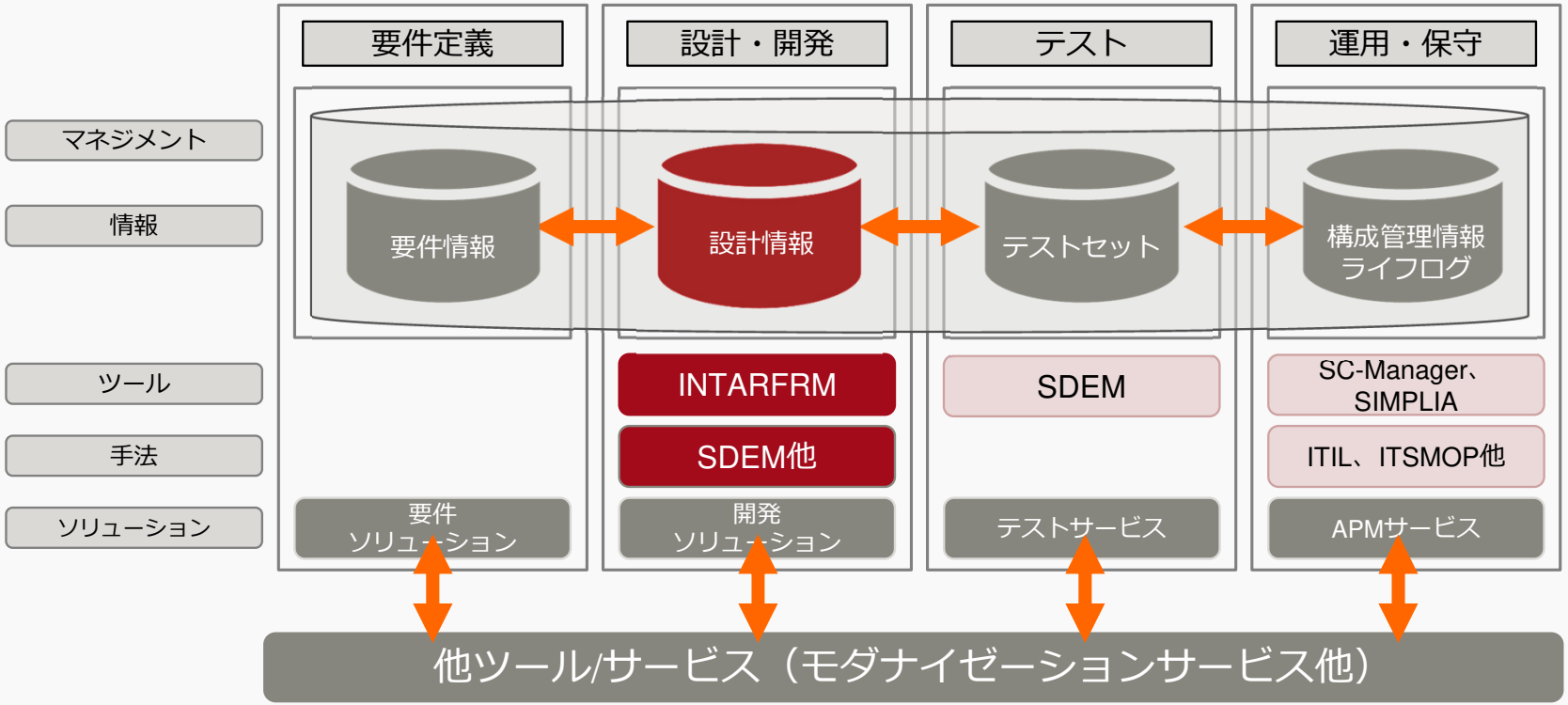
富士通独自の最新技術と技法の体系の中で、INTARFRMはアプリケーションの
LCMを支える富士通標準のアプリケーションフレームワークです。



システムのLCMを支える INTARFRM

FUJITSU

工程毎に最適な手法、ツールをご提供し、全ての工程を支えます。

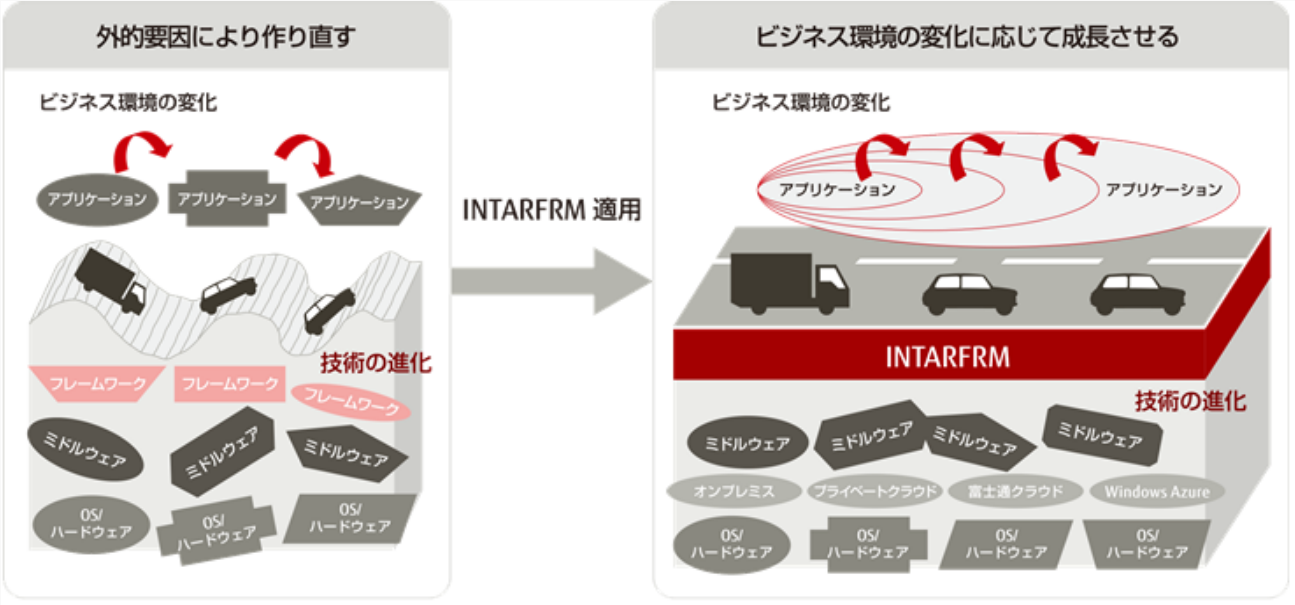


INTARFRM を軸にしたシステムのLCMを実現
要件定義から運用・保守まで富士通が全面的にサポート

アプリケーションフレームワークの必要性１

FUJITSU

OS、ミドル、ハードウェアなどのインフラに依存しないアプリケーションの成長と
長期利用を実現できる それがアプリケーションフレームワークです。

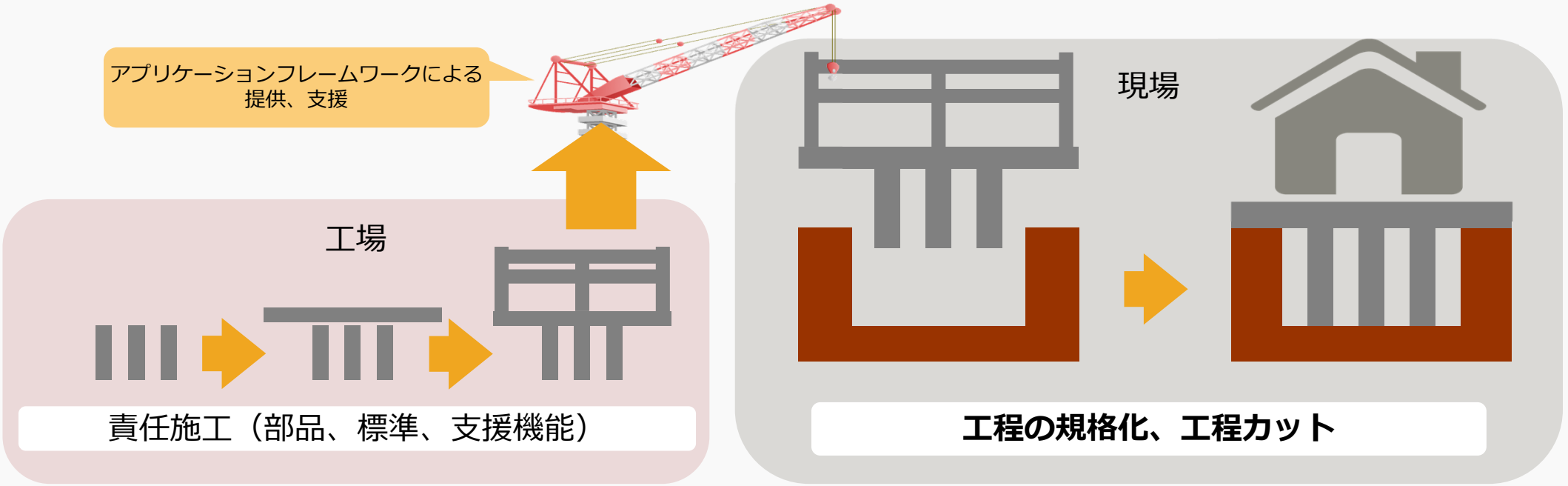


これからはアプリケーションフレームワークで
ビジネス環境の変化に応じてアプリケーションを成長させる時代

アプリケーションフレームワークの必要性２

FUJITSU

アプリケーションのベースとなる部品群、標準規約や、支援機能による作らない開発で、
短納期・高品質なシステム構築が実現できる、それがアプリケーションフレームワークです。



耐震も骨組みも工場での責任施工。
現場では、規程された工法に従い、作業工程もカット

アプリケーションフレームワークの主な機能

FUJITSU

アプリケーションフレームワークの主な提供内容

■アプリケーションの実行機能

■設計作業・開発作業の支援機能

■開発作業プロセスの標準化

狭義のアプリケーションフレームワーク
※Struts、Springなど

広義のアプリケーションフレームワーク
※INTARFRM の位置づけ

(*1) OSSとの連携機能を提供

(※)富士通受託開発時のみ

INTARFRMは実行機能だけでなく、幅広く提供する
「広義」のアプリケーションフレームワークの位置づけ

2. INTARFRMの特長

INTARFRMの５つの特長

FUJITSU

1	**ずっと使える** ～ソフトウェアのライフサイクルに対応～	設計情報のリポジトリ※を核とした首尾一貫した手法により、設計から保守にいたるまでのソフトウェアライフサイクルに対応し、アプリケーションを長期に成長させるお手伝いをします ※リポジトリ・・・システムやアプリケーションの情報をまとめて保管するデータベース
2	**いつでもどこでも開発できる** ～インターネット環境への対応～	時間と場所の壁を越えた開発環境が離れた場所の人々をつなぎ、 設計工程の品質を飛躍的に向上させ、開発期間の短縮が可能です
3	**さまざまな条件下で動かせる** ～最新技術への対応～	オンプレミスでの運用だけでなく、クラウド（SaaS）環境での運用にも対応します。 SOAの適用により他システム連携を容易にします
4	**いろいろ選べる** ～幅広い言語・アーキテクチャーに対応～	設計や保守のニーズに合わせた開発スタイル（ライフサイクル重視型、機動力重視型）をご提案します。また、複数の開発言語やシステム形態を選択することができます
5	**みんなが使える** ～当社グループのノウハウを お客様のプロセス改善に～	開発言語やシステム形態に依存しない開発ノウハウと開発スタイルを確立しています。また、富士通グループのノウハウを結集した標準化・作業プロセスを整備しプロセス改善をお手伝いします

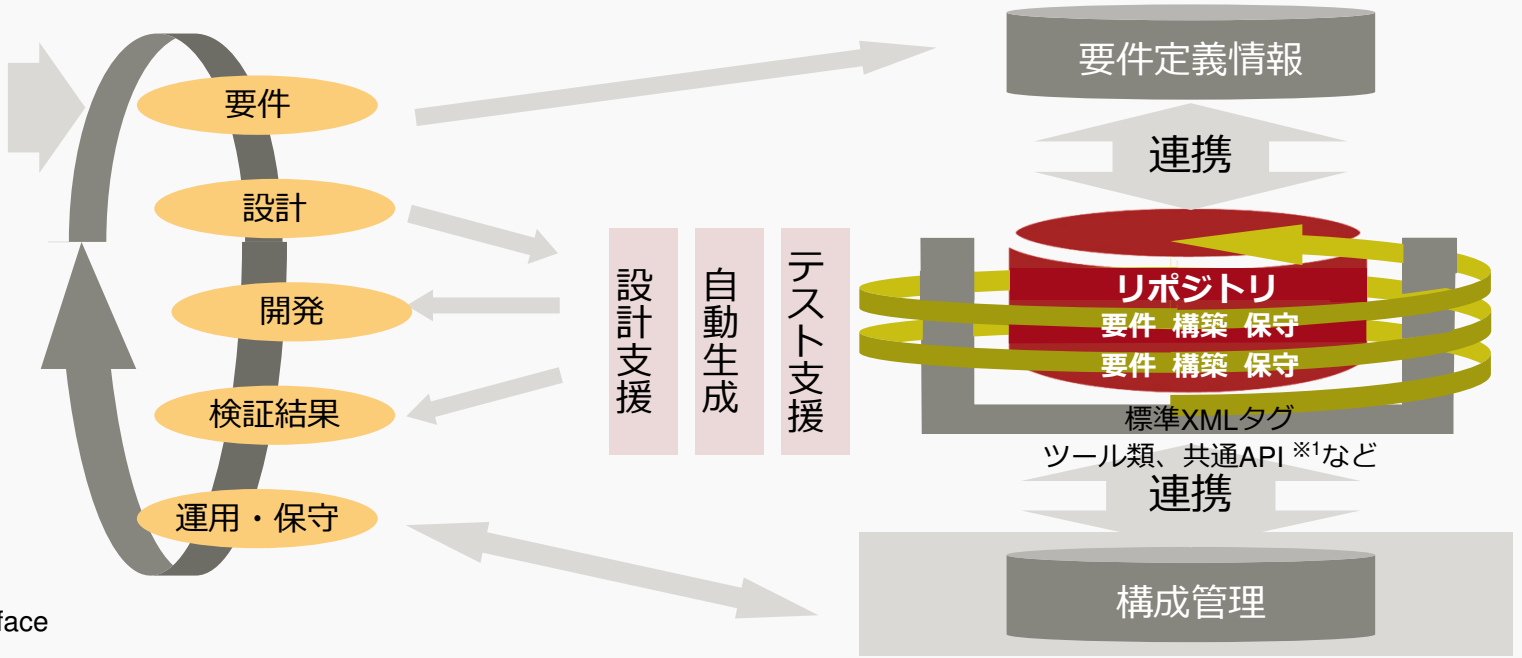
お客様のビジネスと共に進化する
ICTシステムを支えるアプリケーションフレームワーク

①ずっと使える　～ソフトウェアのライフサイクルに対応～

課題：要件反映漏れや要件・設計変更作業での影響箇所の把握を効率化できない
施策：リポジトリを活用して、要件および設計のトレーサビリティを向上する
- 「新要件定義手法」により経営の目的に合致した要件が実装されていることの確認を支援する
- ソースコードやテスト仕様書を自動生成し、ソフトウェアの品質向上・効率化を支援する
- 要件変更・設計変更・保守改修時に、影響箇所の判別を容易にする

要件定義手法

- 要求形成手法：
- 業務形成手法：
- 業務仕様形成手法：

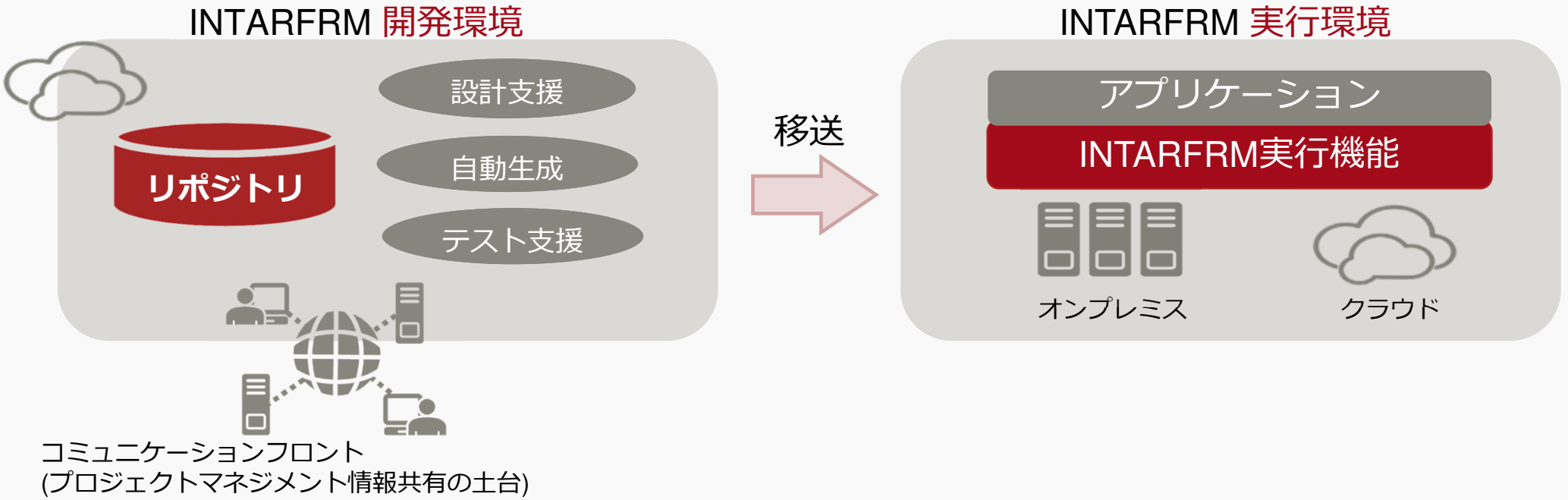


※1 API・・・Application Programming Interface

②いつでもどこでも開発できる　～インターネット環境へ対応～

FUJITSU

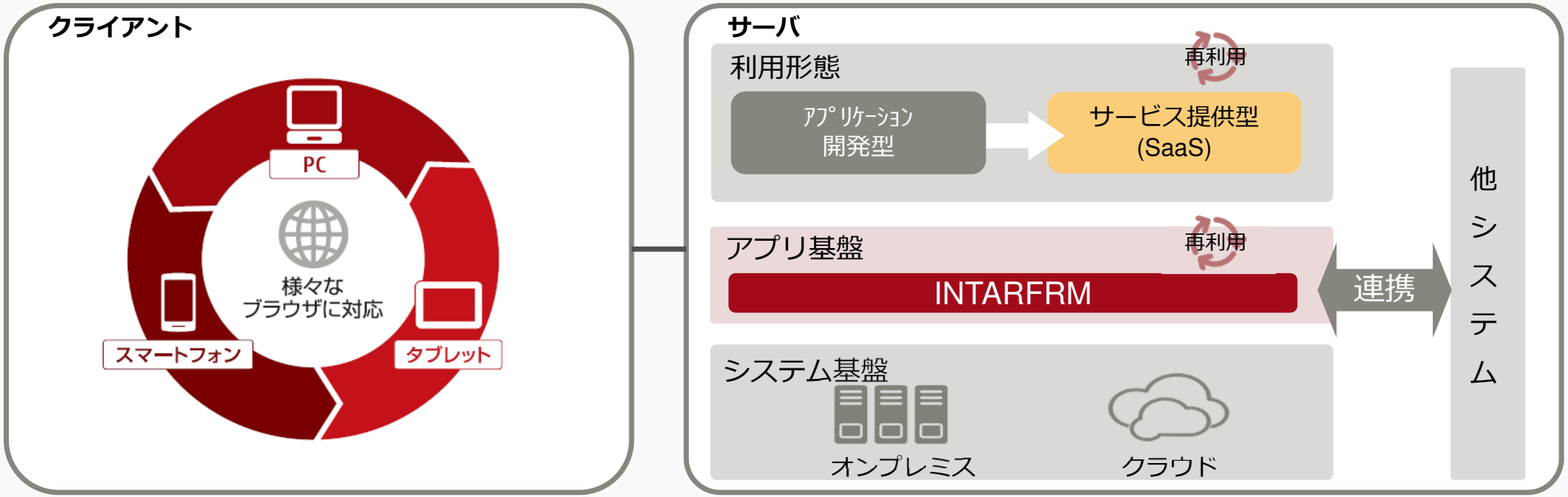
課題：大規模開発やオフショア開発において、設計書や開発資産の集約・整合を
　　手作業に頼らざるを得ないため、分散開発の効率化を阻害している
施策：開発環境をインターネットに対応させ分散開発を支援する
■開発環境をクラウド上に配置し、いつでもどこでも開発可能にする
■クラウド上で開発したアプリケーションを実行環境へ移送して動作可能にする



コミュニケーションフロント
(プロジェクトマネジメント情報共有の土台)

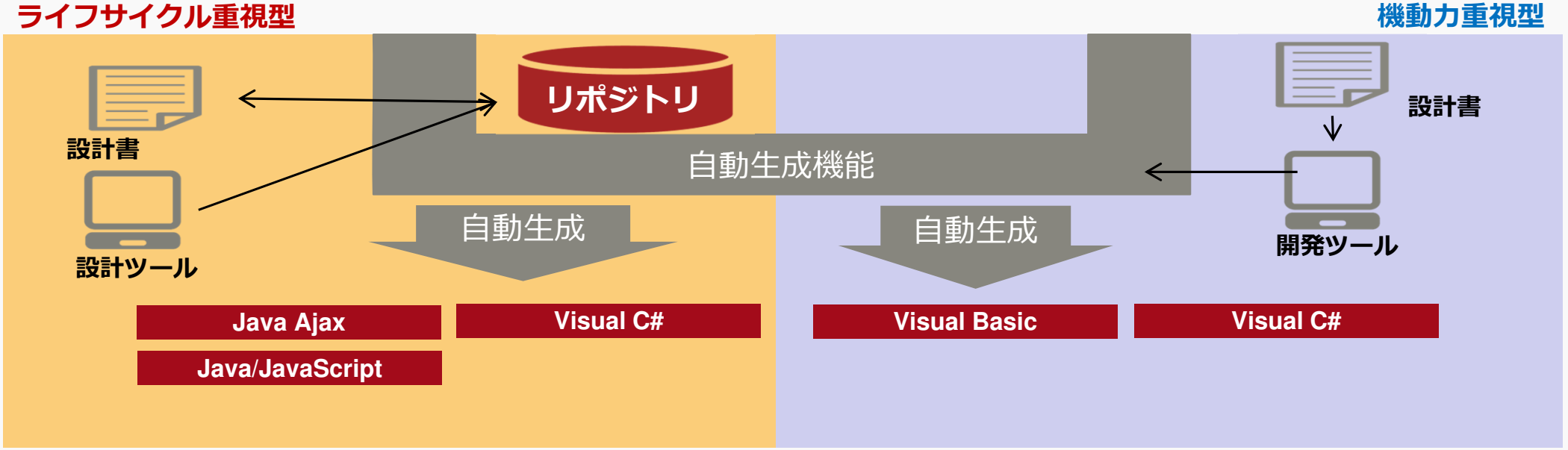
③さまざまな条件下で動かせる　～最新技術への対応～

課題：アプリケーション開発型・サービス提供型など利用形態に合わせた開発をする必要がある
　　時代に合わせて最新のクライアント技術を取り込む必要がある
施策：■アプリケーションのSaaS化を容易にする
　　■SOAの適用により他システムとの連携を容易にする
　　■マルチデバイス・マルチブラウザでの動作を可能にする



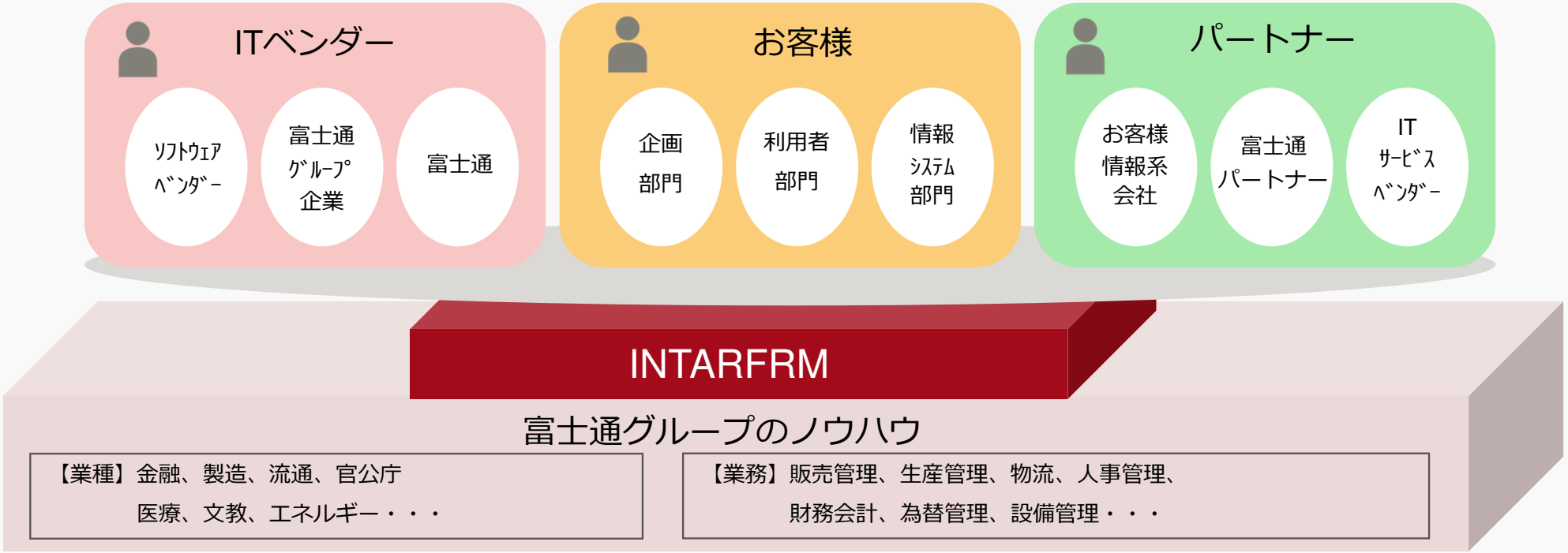
④いろいろ選べる　～幅広い言語・アーキテクチャーに対応～

課題：レガシーシステムの資産を活用できない
　　開発言語・開発スタイルに合わせたアプリケーションフレームワークを利用しなければならない
施策：■ｼｽﾃﾑ形態：Web、スマートクライアント
　　言語　　：Java、 Visual C# 、Visual Basic
　　■OS/ﾐﾄﾞﾙｳｪｱなどの技術変化を吸収し、長期にｱﾌﾟﾘｹｰｼｮﾝを成長させ続ける
　　■「ライフサイクル重視型」と「機動力重視型」の開発スタイルを適用可能にする



⑤みんなが使える　～当社グループのノウハウを結集～

課題：アプリケーションフレームワークが複数存在し使い方・作法が異なるためスキルを活かせない
　　異なるアプリケーションフレームワーク間では開発資産の再利用ができない
施策：■富士通グループのノウハウが蓄積されたアプリケーションフレームワークを統合・発展
　　■継続的なノウハウ・スキル・要員の活用が可能

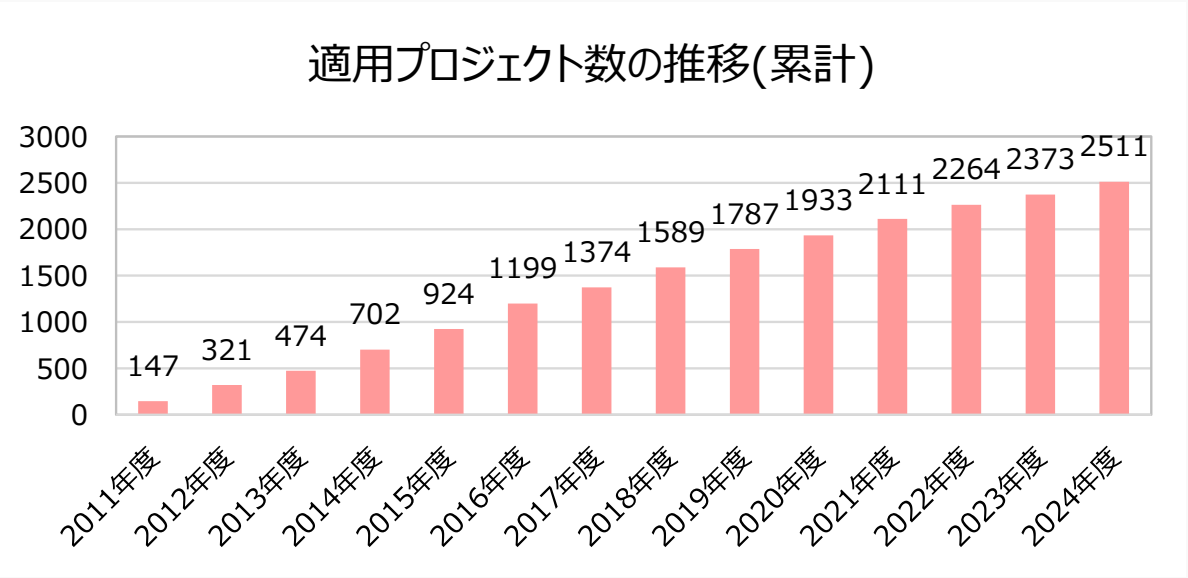


3. INTARFRM適用実績

INTARFRM適用実績

< 2011-2024年度販売・適用実績より >

- 累計適用プロジェクト数 2,511

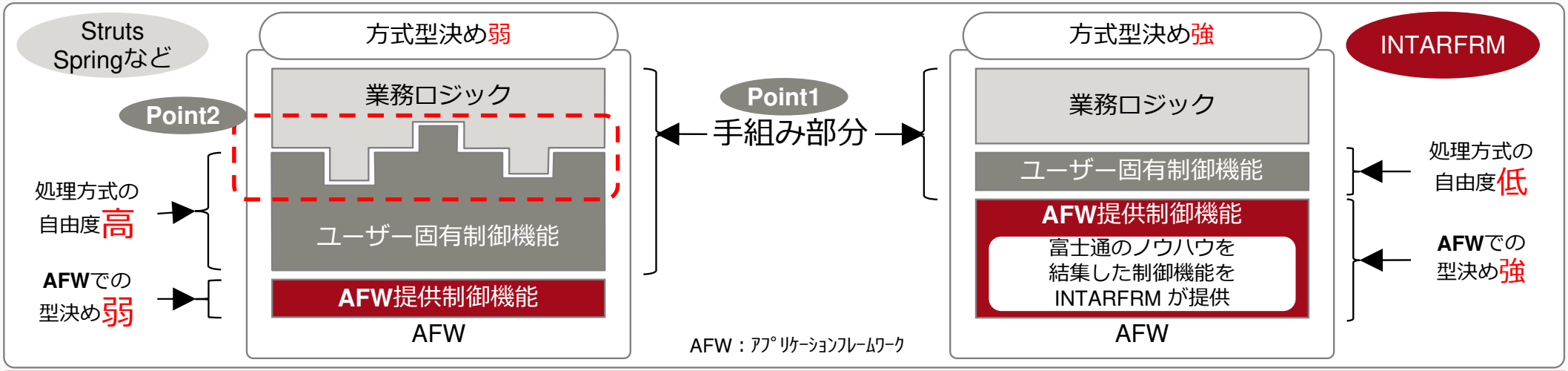


- 適用パッケージ／ソリューション例
  - GLOVIA smart PRONES（中堅製造業向け生産管理システム）
  - GLOVIA Process C1（プロセス産業向け基幹業務パッケージ）
  - Tomorrowchain（量販店向けパッケージ）
  - tsClinical（ 臨床試験（治験）・製造販売後ソリューション）
  - E3CIS（電力小売事業者向け顧客管理・料金計算パッケージソリューション）

4. INTARFRM適用効果

アプリケーション品質向上への効果

INTARFRM はアプリケーション処理方式を型決めし、業務ロジックと制御ロジックを分離。
また、提供部品、自動生成により、手組み部分を削減し、品質、保守性の高いアプリケーションが構築できます。



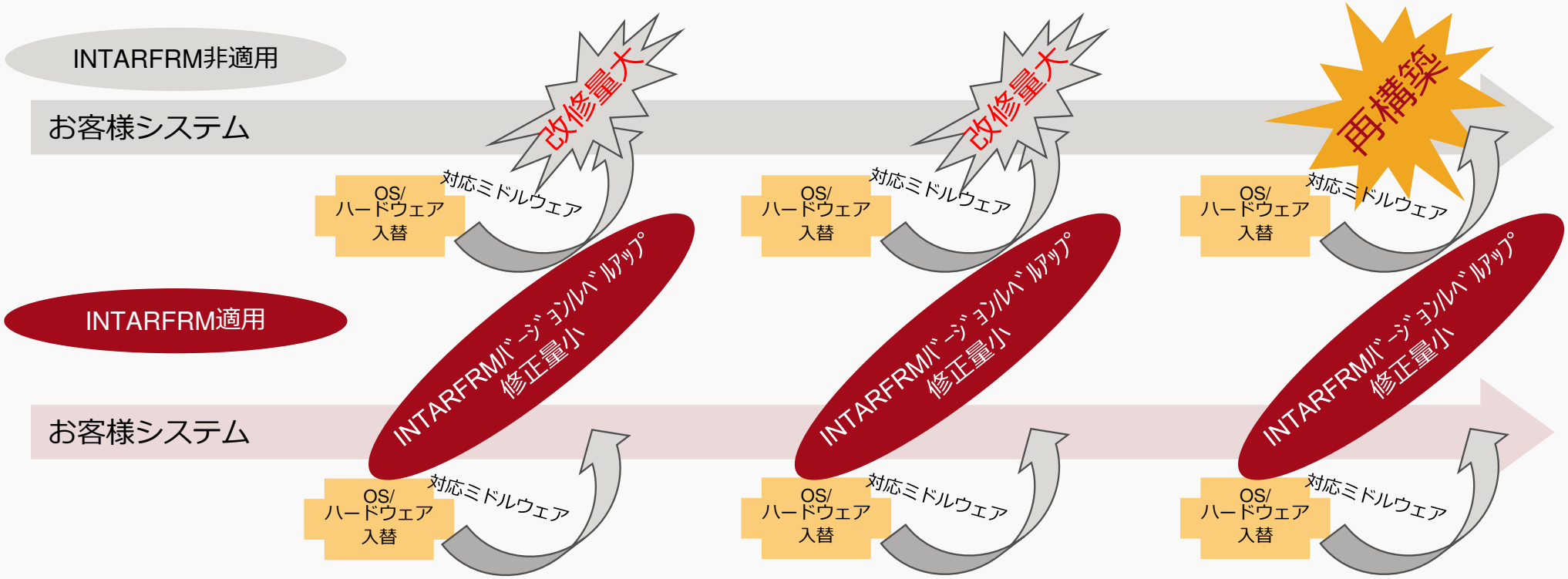
Point1 アプリケーション処理方式の型決めによる効果

・方式の型決め、提供部品、自動生成機能により、業務ロジック部分の開発が主となるため、
開発・試験・保守対象資産が少なくなります。

Point2 業務ロジックと制御ロジックの分離による効果

・業務ロジックと制御ロジックを分離し、結合度を弱くすることで、初期開発だけではなく保守作業や
追加案件開発においてもコストを低減できます。

アプリケーションの成長を支える
～ハードウェア(OS,ミドル)更改の場合～



OS,ミドル技術最新化への対応による効果

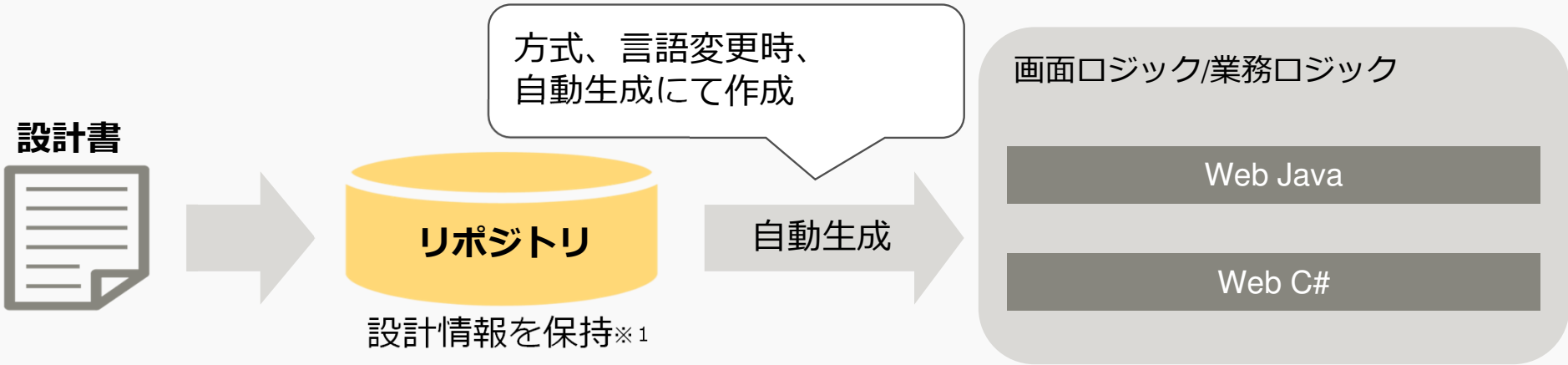
INTARFRM実行機能のバージョンアップを行うことにより対応します。
アプリの修正箇所を抑えることが可能。(※)
そのため、更改対応に煩わされることなく、保守費用を新規事業対応費用に振り向けることが出来ます。

※新技術対応の場合、INTARFRM適用でもアプリ修正が発生する可能性があります

アプリケーションの成長を支える
～フロントエンドのアプリケーション変更の場合～

【フロントエンド・バックエンド一体型】

フロントエンドは技術革新が速い領域です。
フロントエンドの技術革新の変化に、早期に対応できます。



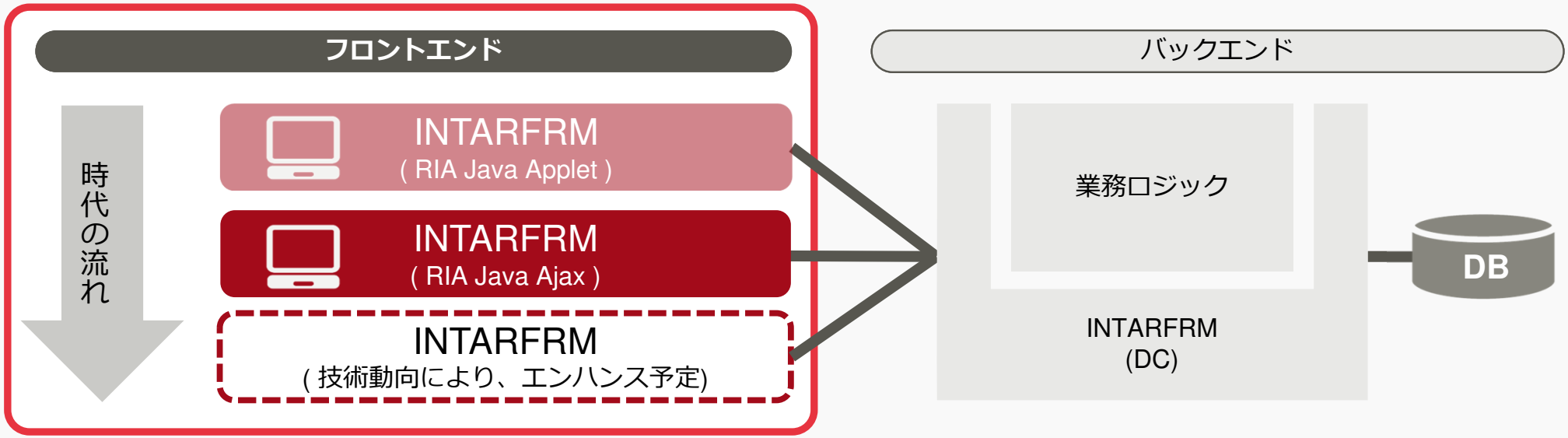
※1　業務ロジックの設計情報は保持せず、手組みになります。

フロントエンドでの効果

INTARFRMのリポジトリの設計情報は言語に依存せず原本を管理します。
業務要件からC#の画面をJava Servletに変更する場合は、リポジトリから自動生成を行うことで、
ほぼ全ての画面層のアプリを作り直すことが可能です。　※レイアウトは別途作成必要

アプリケーションの成長を支える
～フロントエンドのアプリケーション変更の場合～

【フロントエンド・バックエンド分離型】
フロントエンドの最新技術に追従できます。

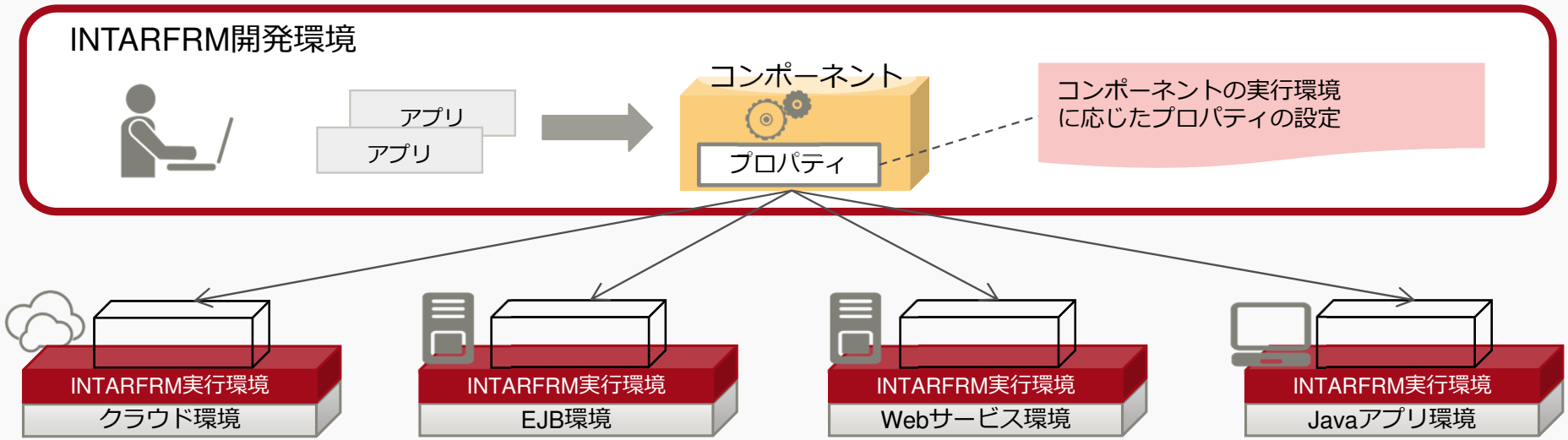


フロントエンドでの効果

・最新技術に対応することができます。
・今後、次世代技術に追従する場合、バックエンドには影響を与えず、フロントエンドのみの対応で可能となります。

アプリケーションの成長を支える
～バックエンドの実行環境変更の場合～

【フロントエンド・バックエンド分離型】
バックエンドは長期利用したい部分です。
アプリケーションのコンポーネント化により、実行環境の変更や他システムとの連携などに、柔軟に対応できます。



バックエンドでの効果

INTARFRMのフレームワーク機能とプロパティ（環境定義や制御アプリ）の変更により、実行環境の変更に関わらず、業務ロジックの修正を不要とします。また、他システムの業務コンポーネントとの連携を可能にします。そのため、変更作業期間の短縮を見込めます。

5. オープンソースソフトウェアとの比較

オープンソースソフトウェア（ＯＳＳ）との違い

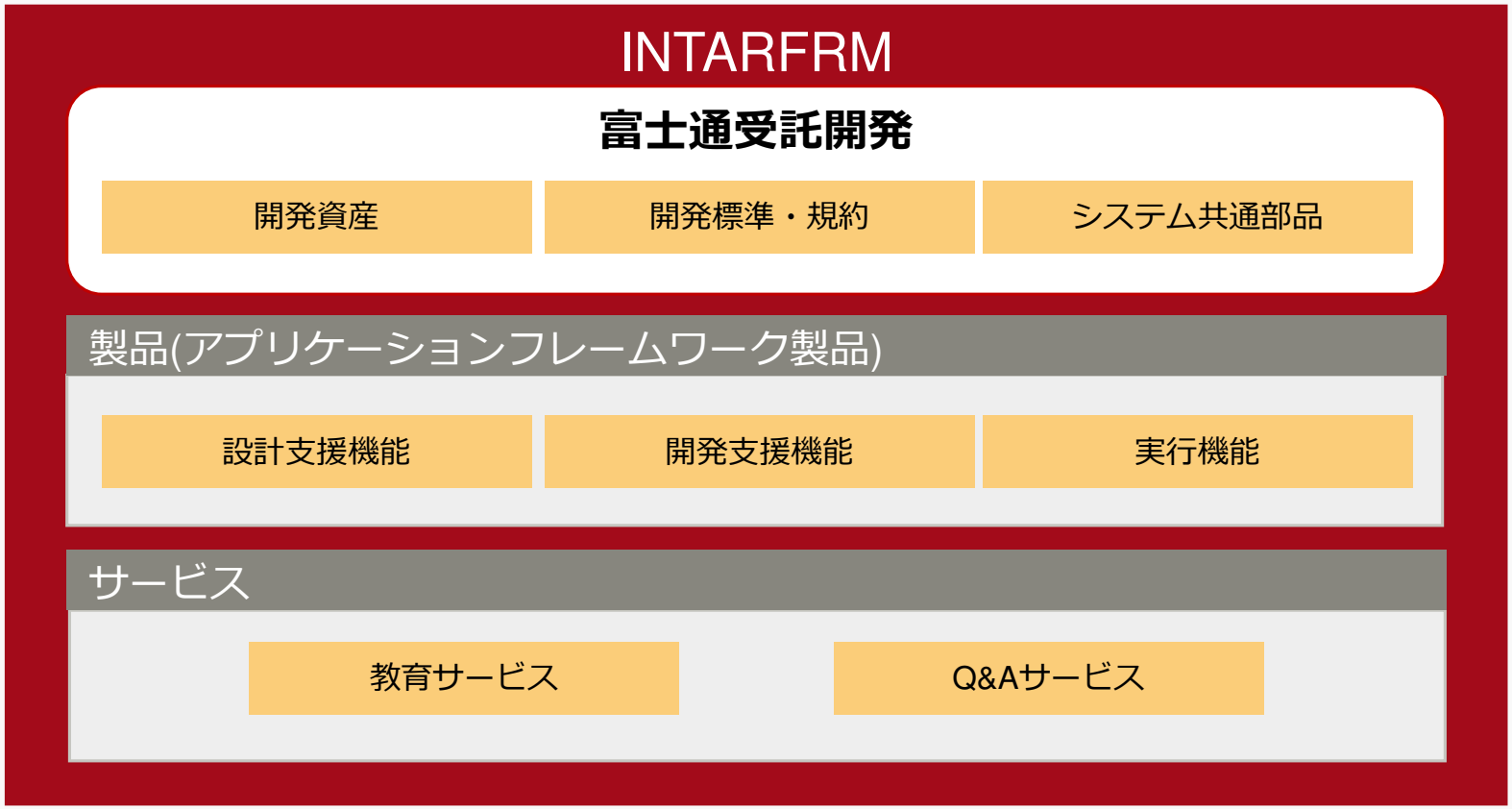
FUJITSU

	ＯＳＳ	INTARFRM
サポート体制	ベンダのサポートがないため、 お客様自らが全てのリスクをとり、 障害対応、ＱＡ対応、方式設計まで 行うことになります。	富士通が提供するアプリケーション フレームワークであるため、 富士通の全面的なサポートをご提供。 安心してご利用いただけます。 
サポート範囲	ＯＳＳアプリケーションフレームワークは 基本実行制御だけのサポート。 全工程を対象としていないため、 工程毎に必要なツールの組合せなど お客様の検討となります。	富士通の今までのノウハウ、手法を元に、 全工程を対象としたサポートが可能です。 実行制御だけでなく、各工程のプロセスの型決めで、 お客様のプロセス改善のお手伝いをいたします。 
プロジェクト対応要員	スキル保有者が多いが、 お客様がスキルを見極めて確保する 必要があります。	富士通グループ、ビジネスパートナーへの教育を 用意しています。 

6. INTARFRMの提供資材

INTARFRMの提供資材

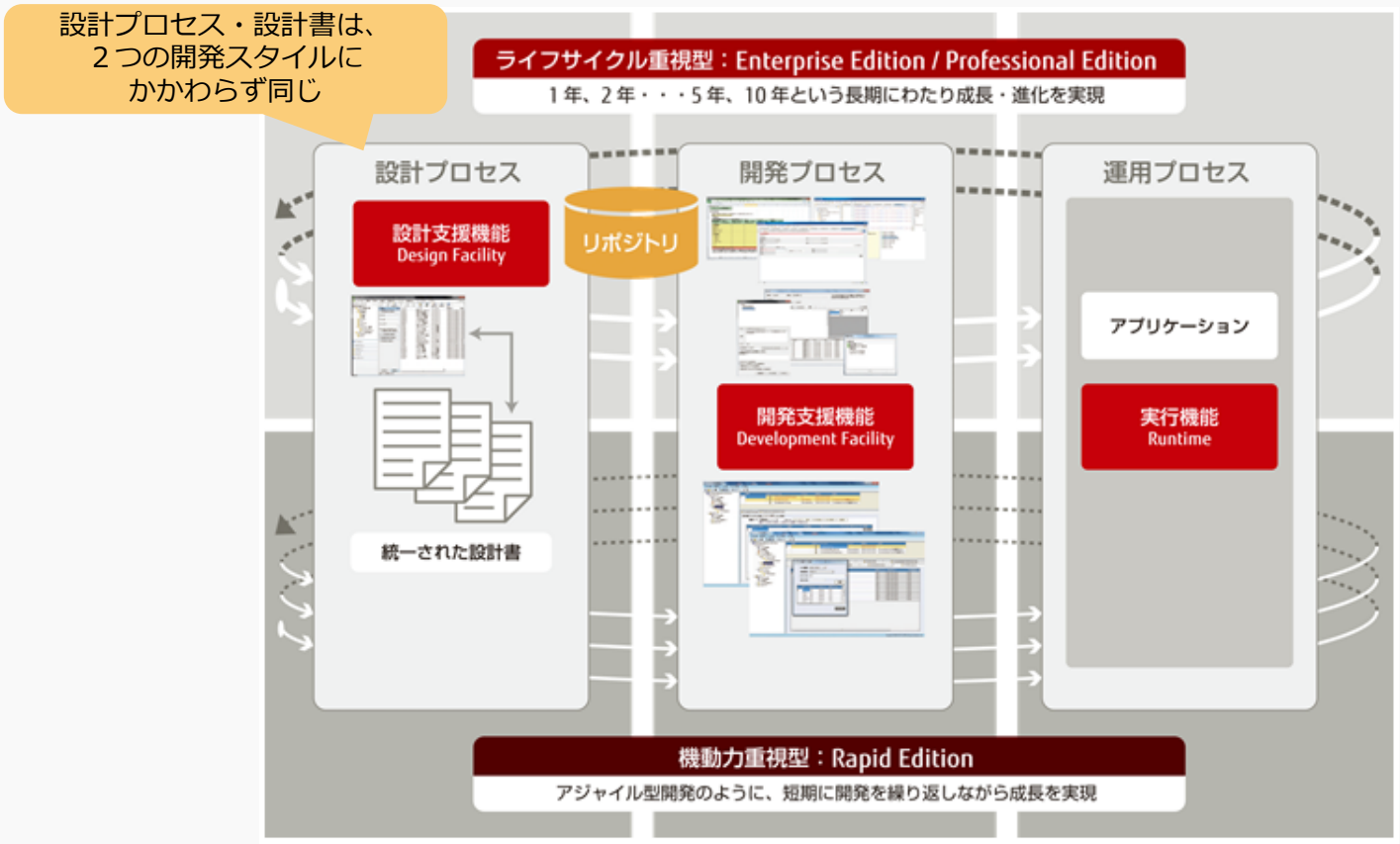
INTARFRM は製品・サービスに加え、富士通受託開発の場合は、富士通グループのノウハウを結集した作業プロセスを提供します。この標準化された作業プロセスを用いることで、 お客様のプロセス改善にも寄与します。



7. アプリケーションフレームワーク製品

標準化された作業プロセス

INTARFRMは、標準化された作業プロセスと、システムの成長サイクルに注目した２つの開発スタイルを備えています。
システムの形態や開発言語が異なっていても、同じプロセスで開発できます。



標準化された作業プロセスを支える3つの製品群

FUJITSU

システムを開発・運用する中で、何を重視するかによって、開発スタイルを選択できます。
開発スタイルに関わらず設計プロセス・設計書は同じです。
また、INTARFRMは「設計支援機能」、「開発支援機能」、「実行機能」の3つの製品群で構成されます。
3つの製品群はそれぞれ連携し、システムの設計、開発、運用、保守を支援します。

ライフサイクル重視型 Enterprise Edition/Professional Edition

設計支援機能　開発支援機能　実行機能

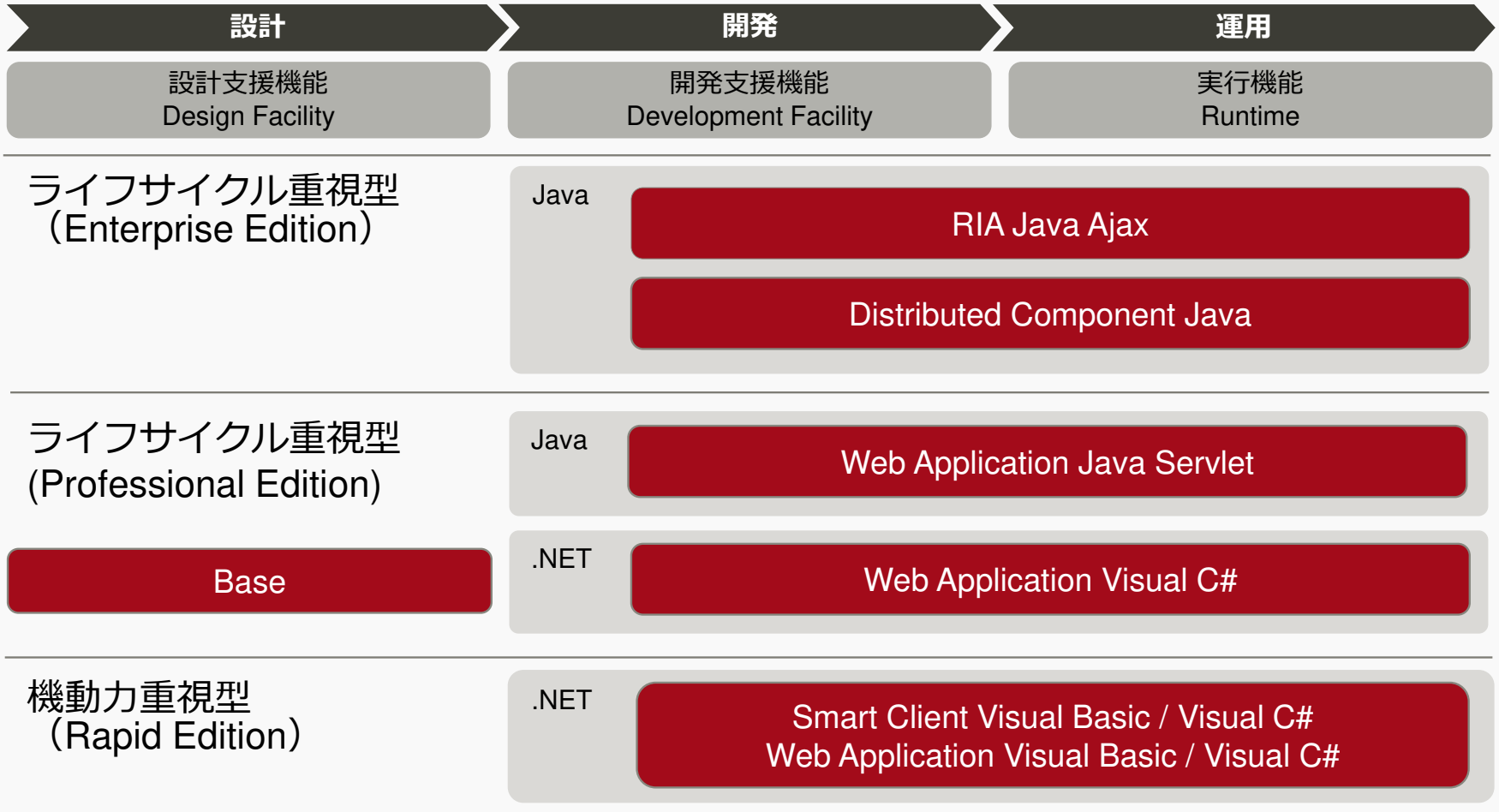
長期の資産維持を重視・・・設計情報をリポジトリやSDEMの設計書によって管理することで、ソフトウェアのライフサイクル全般にわたって資産（設計ドキュメント、ソースコードなど）の品質を維持する開発スタイルです。

機動力重視型 Rapid Edition

設計支援機能　開発支援機能　実行機能

短期に繰り返す成長を重視・・・設計情報をSDEMの設計書によって管理し、要件から設計・開発・ 評価を短期間に繰り返すことによって資産（設計ドキュメント、ソースコードなど）を順次成長させる、機動力を重視した開発スタイルです。

【ご参考】INTARFRM全体の製品体系



8. 参考資料

<参考> INTARFRMの開発スタイル(1/3)

FUJITSU

開発スタイル毎の特徴は以下の通りです。

<table>
<tr><th colspan="2"></th><th>ライフサイクル重視型
(Enterprise Edition)</th><th>ライフサイクル重視型
(Professional Edition)</th><th>機動力重視型
(Rapid Edition)</th></tr>
<tr><td colspan="2" rowspan="2">特徴</td><td colspan="2">ソフトウェアのライフサイクル全般にわたって資産（設計ドキュメント、ソースコードなど）の品質を維持する開発スタイル。長期的な運用保守を考える場合に選定する。
Professional Editionではリポジトリの利用が可能。</td><td rowspan="2">設計情報をリポジトリ管理せず、
要件から設計・開発・評価を短期間に
繰り返すことによって、資産(設計ドキュメント、ソースコードなど）を順次成長させる機動力を重視した開発スタイル。
短期開発や小規模～中規模で開発を行いたい場合に選定する。</td></tr>
<tr><td>フロントエンド（画面ロジック）とバックエンド（業務ロジック）の分離により、フロントエンド技術変化の追従やバックエンド資産の長寿命化をより重視したスタイル。</td><td>フロントエンドとバックエンドを一体とすることにより、よりシームレスな開発を重視したスタイル。</td></tr>
<tr><td colspan="2">設計情報の管理</td><td>フロントエンド・バックエンド共に、SDEMの設計書で管理する。</td><td>設計情報（項目、エンティティ、フォームなど）をリポジトリで管理する。
※リポジトリで管理できるのは自動生成の対象となる設計情報のみ。
手組みとなる業務ロジックの設計情報はリポジトリの管理対象外でSDEMの設計書で管理する。</td><td>SDEMの設計書で管理する。</td></tr>
<tr><td rowspan="2">設計／開発</td><td>設計ドキュメント
設計プロセス</td><td colspan="3">２つの開発スタイルに関わらず、設計ドキュメントの書式、設計プロセスは同じであり、INTARFRM標準として標準化されている。
富士通SEはこのノウハウを活用し、品質の良い設計を行える。</td></tr>
<tr><td>自動生成</td><td>フロントエンド・バックエンド共に、
製品で提供される制御機能や定義ファイルを組み込み、システムを開発するスタイル。</td><td>リポジトリで管理された設計情報を元にソースコード、制御部品やテスト仕様書などの自動生成を行う。</td><td>Map comPonent Element Editorにて
アプリケーションで使用される各種設定ファイル、マップファイルの生成、および画面データクラスと画面処理クラスのソースコードの自動生成を行う。
また、ミッドティアのソースコードをRapid Addin for Visual Studioで自動生成を行う。</td></tr>
</table>

<参考> INTARFRMの開発スタイル(2/3)

FUJITSU

		ライフサイクル重視型 (Enterprise Edition)	ライフサイクル重視型 (Professional Edition)	機動力重視型 (Rapid Edition)
設計／開発	業務ロジック	フロントエンド・バックエンド共に、 提供される標準ライブラリの部品を使用して手組みする。 また各種定義情報を外部管理することにより、画面制御/業務制御ロジックのコーディング量を削減することが可能。 なおバックエンドでは、業務ロジックを分散コンポーネント化して開発を行う。	業務ロジックの雛型が自動生成され、業務ロジックは手組み。 コンポーネント再利用機能を利用することで、再利用資産情報（既存のJARファイル、DLLファイルなどを解析して得られるコンポーネント情報）を取り込み、再利用資産を呼び出すようなロジックを自動生成することが可能。	提供される標準ライブラリの部品を使用して手組みする。画面に表示する項目の情報をマップ・コンフィグファイル（XML形式） に登録し、ランタイムが実行時に制御することにより、画面制御ロジックのコーディング量を大幅に削減することが可能。また、ミッドティアのレイヤごとの雛形ソースは自動生成され、業務ロジックは手組で行うことが可能。
	モックアップ	フロントエンドでは、モックアップでの設計レビューが可能。	リポジトリ(設計情報)だけで、擬似的なモックアップで実画面を見ながらの設計レビューが可能。	画面設計時に作成し、レビューで確認した画面ソースを開発工程でそのまま使用することが可能。 モックアップでの設計レビューは不可能。
運用	実行	フロントエンド・バックエンド共に、 画面情報、項目情報、メッセージ情報、 画面遷移情報などを各種定義ファイルで管理し、動的に制御することによって実行する。業務ロジックについては、コンパイル・配備することで実行する。	画面情報、項目情報、メッセージ情報、画面遷移情報などを継承ファイルで管理し、コンパイル・配備することで実行する。	画面情報、項目情報、メッセージ情報などをマップ・コンフィグファイルで管理し、実行時に動的に制御することによって実行する。
	影響検索	RIA Java Ajaxでは、定義情報変換ツールのExcel 検索機能を利用可能。	プロジェクトナビゲータのクロスリファレンス機能を利用し、リポジトリから影響検索を行う。 クロスリファレンス機能では、 ・桁数変更対象の項目を使用しているフォーム、エンティティの検索 ・エンティティを使用しているプログラムの検索などの影響検索が可能。	Map comPonent Element Editorのシステム共通マップ検索機能や逆引き検索機能を利用し、マップファイル間の影響検索を行う。 システム共通マップ検索機能や逆引き検索機能では、桁数変更対象のドメインを使用しているフォームなどの影響検索が可能。 なお、エンティティは検索不可能。

<参考> INTARFRMの開発スタイル(3/3)

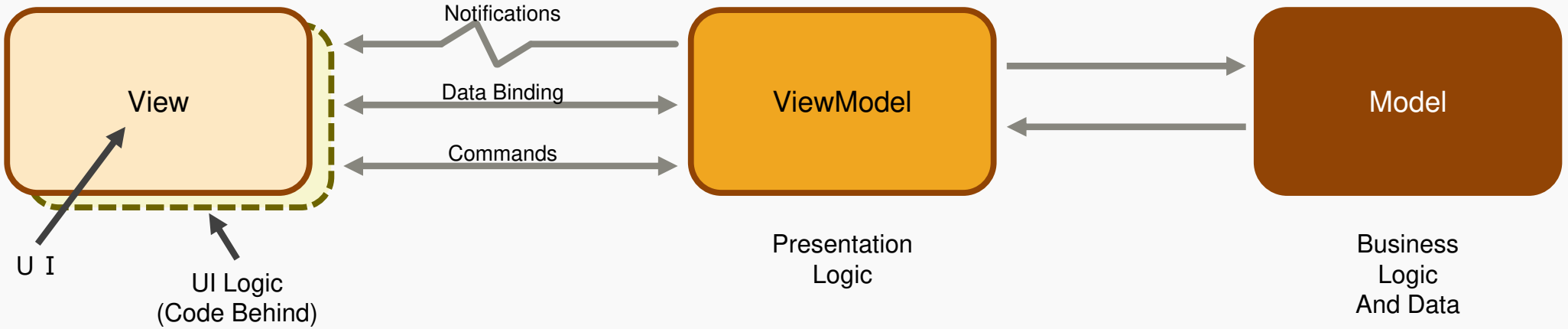
FUJITSU

		ライフサイクル重視型 (Enterprise Edition)	ライフサイクル重視型 (Professional Edition)	機動力重視型 (Rapid Edition)
運用	仕様変更	フロントエンド・バックエンド共に、設計書を修正し、ソースコードの手修正を行う。 仕様変更が入りやすい定義情報については外部管理することで、ソースコードの修正を極力減らし、保守しやすくしている。 なおフロントエンドでは、画面エディタの利用によってレイアウト修正を容易に行うことが可能。またバックエンドでは、分散コンポーネント化により、修正箇所を局所化して修正することが可能。	リポジトリの設計情報を修正し、 ソースコード、設計ドキュメントの自動生成を行い、差分をソースコードに反映する。仕様変更が入りやすい画面項目情報やメッセージ情報などは継承ファイルで管理することで、ソースコードの修正を最小限に減らしている。	設計書を修正し、Map comPonent Element Editorで各種設定ファイル、マップファイルの変更、および各ソースコードの修正を行う。 また、セルフカスタマイズ機能にてレイアウト修正もソースコードを修正することなく、変更が可能。 仕様変更が入りやすい画面項目情報やメッセージ情報などはマップ・コンフィグファイルで外部管理することにより、ソースコードの修正を極力減らしている。
	資産配布	Ajax： 資産配布不要。クライアントには ブラウザがあればよい。 DC： 資産配布不要。	Webアプリケーション： 資産配布不要。クライアントには ブラウザがあればよい。	Webアプリケーション： 資産配布不要。クライアントには ブラウザがあればよい。 SmartClient： クライアントへの資産配布には、 ClickOnceの利用を推奨。

<参考> INTARFRMのアプリ構造

INTARFRMでは、アプリ構造にMVC(Model-View-Controller)パターンを採用しています。
また、INTARFRM Enterprise Edition RIA AjaxやRapid Editionでは
MMVC(Model-Model-View-Controller)（マイクロソフトではMVVM(MODEL-VIEW-VIEWMODEL)）パターンを採用しています。
これにより、View層の近くでロジックを記述する方式に対応できます。

〈参考〉 MVVMパターン



※「Prism Development Guide Chapter 5: Implementing the MVVM Pattern」の「Class Responsibilities and Characteristics」より引用

<参考> INTARFRMの開発概念

FUJITSU

富士通グループの経験によるノウハウを元に、DIやCoCなど業界標準のアーキテクチャーを発展させ、
INTARFRMのアーキテクチャーとして採用しています。

DI （Dependency Injection：依存性の注入）

コンポーネント間の依存関係をプログラムのソースコードから排除し、外部の設定ファイルなどで
注入できるようにするソフトウェアパターン。

- メリット　：プログラム間の結合度低下によるコンポーネント化の促進(保守性の向上)
- デメリット　：使用しすぎると、設定ファイル数や設定内容が増え、システムが煩雑化。
- 代表例　：Spring、Seasar2など

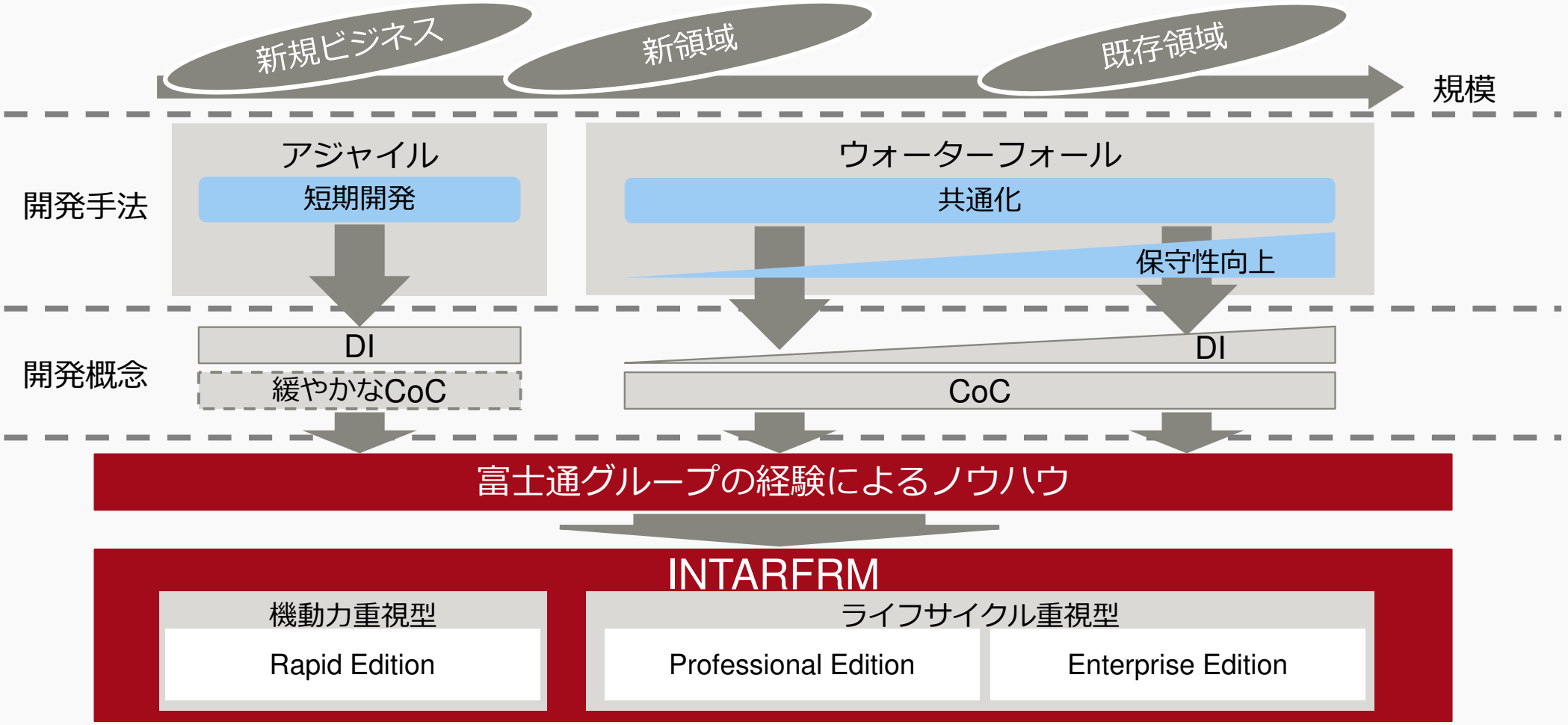
CoC（Convention over Configuration：設定より規約）

開発者の決定すべきことを減少させ、単純にするが柔軟性は失わせないというソフトウェア設計パラダイム。
近年のフレームワークに多い。

- メリット　：規約に則ることによる重複コードの抑制など、開発上での省力化が可能。
- デメリット　：自動化されている部分がブラックボックスとなり、再開発や新領域での
  システム開発などで、既存システムを再利用することが難しくなる。
- 代表例　：Ruby on Railsなど

富士通のノウハウを集結し、
DI,CoCのメリットデメリットをバランス化したアーキテクチャー

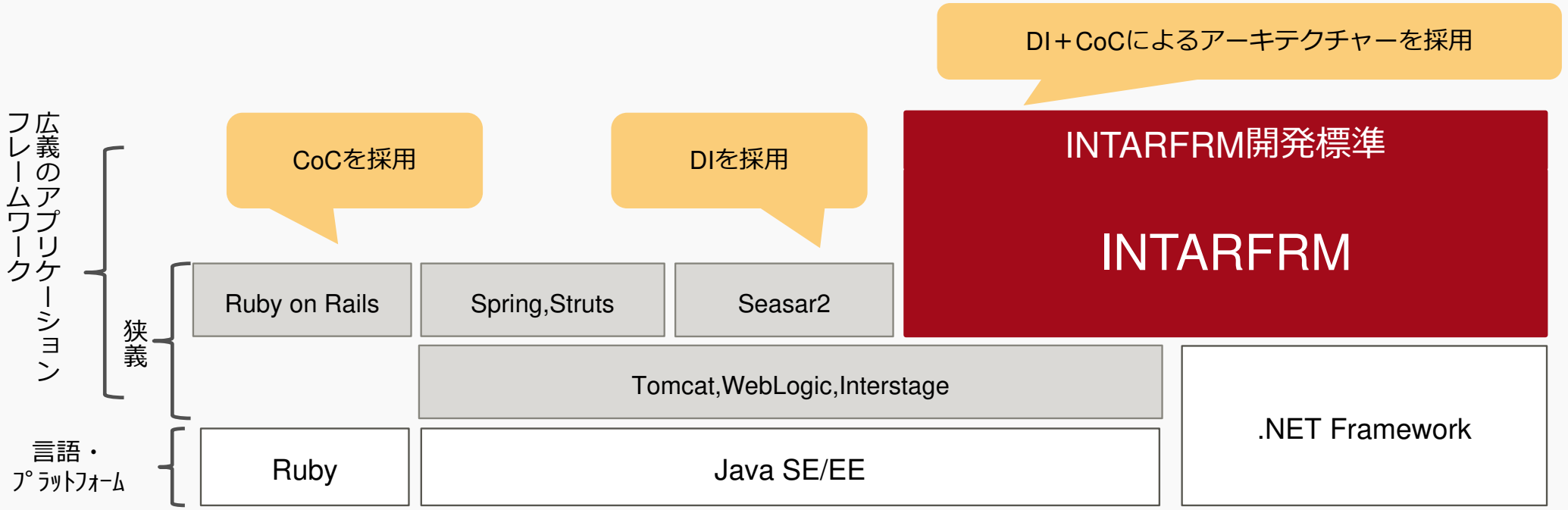
<参考>アーキテクチャーの最適パターンに合った製品群



<参考>フレームワークとしてのINTARFRMの位置づけ

FUJITSU

INTARFRMは、他のフレームワークと比較した場合、以下の位置づけとなっています。



INTARFRMはJavaや.NET全てに対応する
アプリケーションフレームワーク

商標について

FUJITSU

＊INTARFRM、Interstageは富士通株式会社の登録商標です。
＊Visual C#、Visual Basic、.NET Framework、Windowsは、米国 Microsoft Corporationの米国およびその他の国における登録商標または商標です。
＊Smart Clientはマイクロソフト社の技術用語です。
＊Oracle、Java、 WebLogicは、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。
＊Tomcatは、Apache Software Foundationの登録商標または商標です。
＊文中の社名、商品名等は各社の商標または登録商標である場合があります。
＊その他、本資料に記載されている内容には必ずしも商標表示（TM、®）を付記しておりません。

Thank you

FUJITSU