

2021/6/9
MATLAB EXPO 2021

事業推進のための 技術基盤再構築を可能にするMBD



グローリー株式会社

開発本部 デジタルソリューション開発統括部 IoTアプリケーション開発部

中川 誠



- 会社紹介
- MBD導入の背景
- 導入初年度の成果・課題
- 製品適用に向けた活動
- まとめ

事業内容

通貨処理機、情報処理機、
自動販売機、カードシステム、
サービス機器などの
開発・製造・販売・メンテナンス

創業

1918年（大正7年）3月

資本金

128億 92百万円

本社所在地

兵庫県姫路市

従業員数

11,012名（単体：3,503名）

（2021年3月31日現在）

連結業績

売上高 2,174億円 営業利益 142億円

（2020年度）

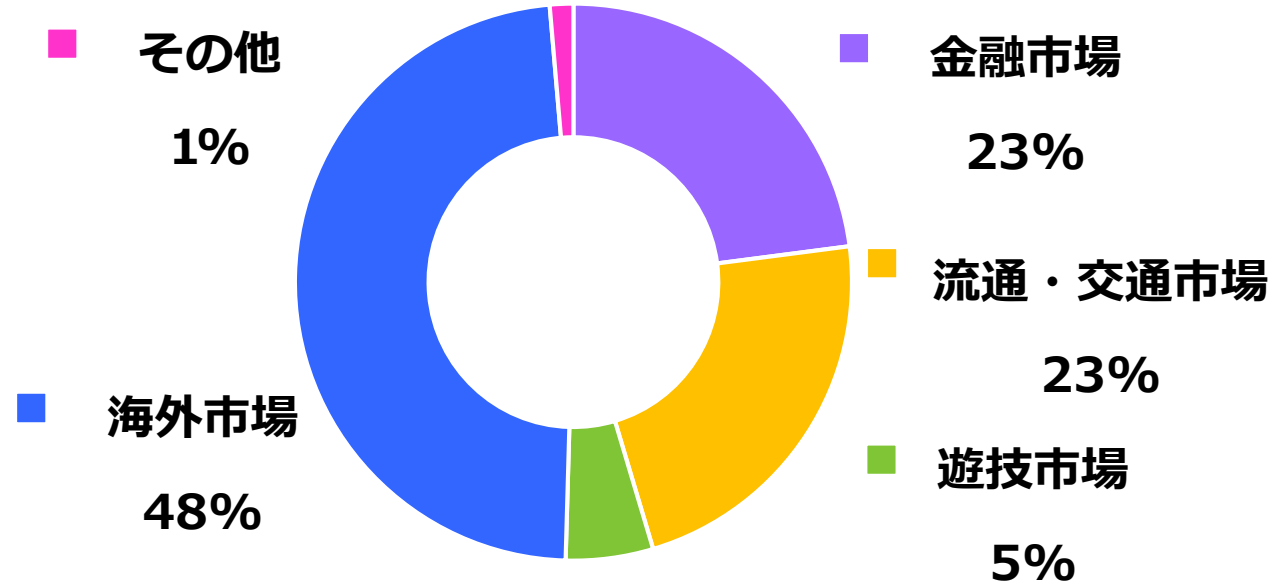


国産第1号の
硬貨計数機



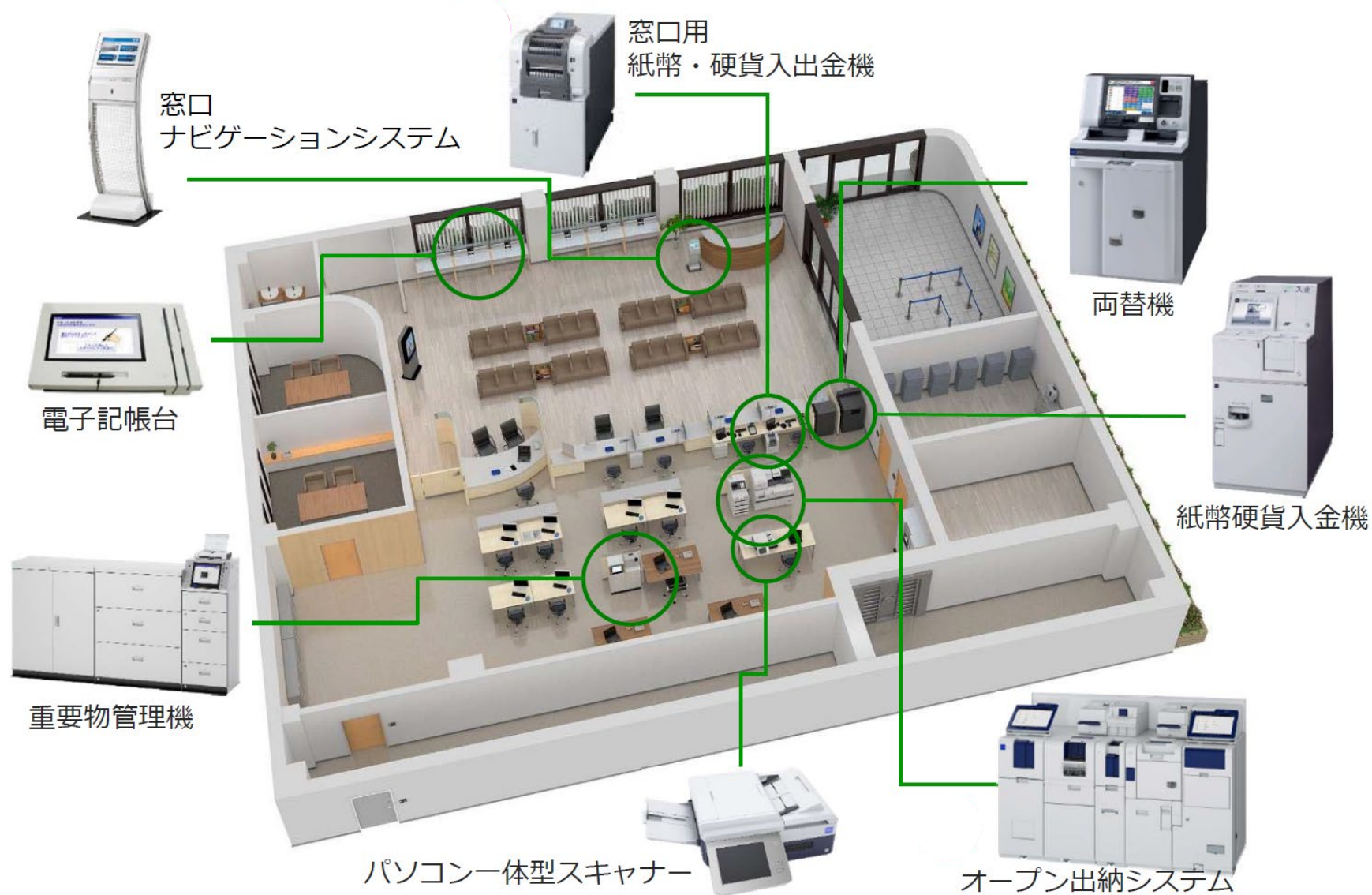
代表取締役社長
三和 元純

事業領域



金融	流通・交通	遊技	海外	顔認証システム/ ロボットSI
				
主な販売先： 金融機関 OEM先	主な販売先： スーパーマーケット、百貨店 警備輸送会社 鉄道会社、病院 たばこメーカー、自治体	主な販売先： 遊技場 (パチンコホール等)	主な販売先： 海外の金融機関 警備輸送会社 小売店、カジノ OEM先	主な販売先： マンション、ホテル、 化粧品製造メーカー

金融市場 金融機関向け主力製品



オープン出納システム



重要物管理システム



銀行内での現金・重要物の厳格な管理をシステムによって実現

海外市場 流通店舗向け主力製品



効率化や省人化を実現するソリューション提案を強化
流通事業を更に加速



シェア

No.1

通貨処理機のリーディングカンパニーとして
国内外でシェアNo.1の実績を誇っています。

金融



オープン出納システム



窓口用紙幣硬貨入出金機



海外向け窓口用紙幣入出金機

スーパー・コンビニ

駅



つり銭機



売上金入金機



ロッカー

※いずれも当社調べ

2021年3月現在

3つの強み

1 研究開発力

業界初の製品・サービスを開発

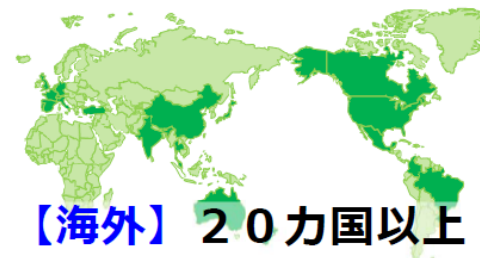


【認識・識別技術】 通貨の真偽や正損を見分ける

【メカトロ技術】 通貨をすばやく正確に数える・仕分ける・束ねる

2 直接販売・直接メンテナンスのネットワーク

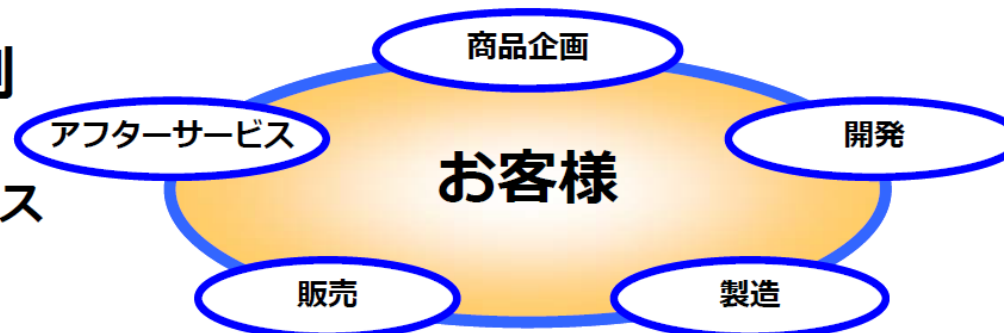
お客様ニーズに応えるご提案と
迅速なサービスを実現



【国内】 110ヶ所以上 【海外】 20カ国以上

3 グループ一貫体制

商品企画からアフターサービス
までグループ内で完結



弊社がこれまでに社会に送り出してきた、お金を「見分ける」「数える」「束ねる」などの機能はコア技術である「認識・識別技術」と「メカトロ技術」から成り立っています。

① 認識・識別技術



・ 紙幣識別技術



・ 硬貨識別技術



・ 帳票判別技術

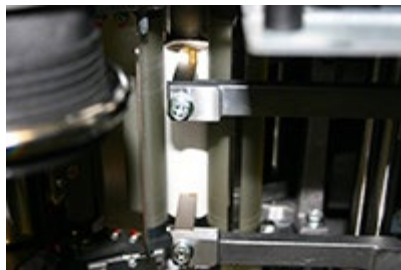


・ 手書き文字認識技術

② メカトロ技術



・ 硬貨選別技術



・ 硬貨包装技術



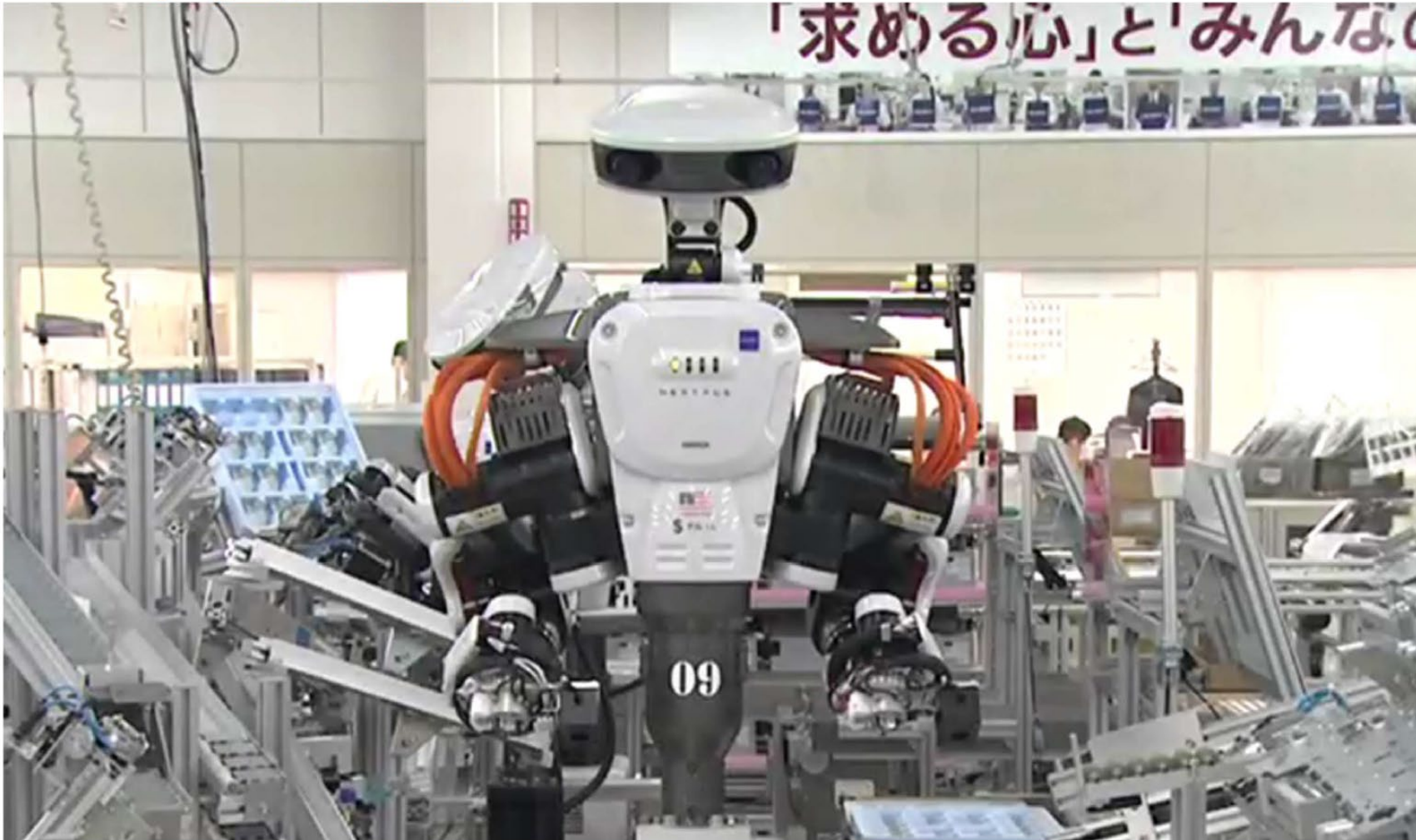
・ 紙幣帯封技術



・ 紙幣整理(表裏反転)技術

ASROF

～ ASROF (アスロフ) ～
Automation Smart Robot for Future



新事業戦略（個体認証）

日本初！大学構内で顔認証決済サービスの実証実験を開始



顔認証による手ぶら決済サービス
「BioPay（バイオペイ）」
に関する実証実験を開始しました。

・ BioPay

「顔画像」と「クレジットカード情報」を登録することにより、支払い時はタブレット端末に「顔」をかざすだけで、現金やクレジットカードの受け渡しをすることなく決済可能となります。



新たなビジネスモデルへの展開



データ化・分析・施策連携まで一気通貫で実現

センサー・
デジタルタッチポイント

AI・ビッグデータ

店舗・人物理解に基づいた
施策連携を実現



IoT・AIでリアル行動を科学し、クライアントのDXを推進

新型コロナウイルス感染症対策

券売機や診療費支払機のタッチレス操作に対応

タッチレス操作イメージ



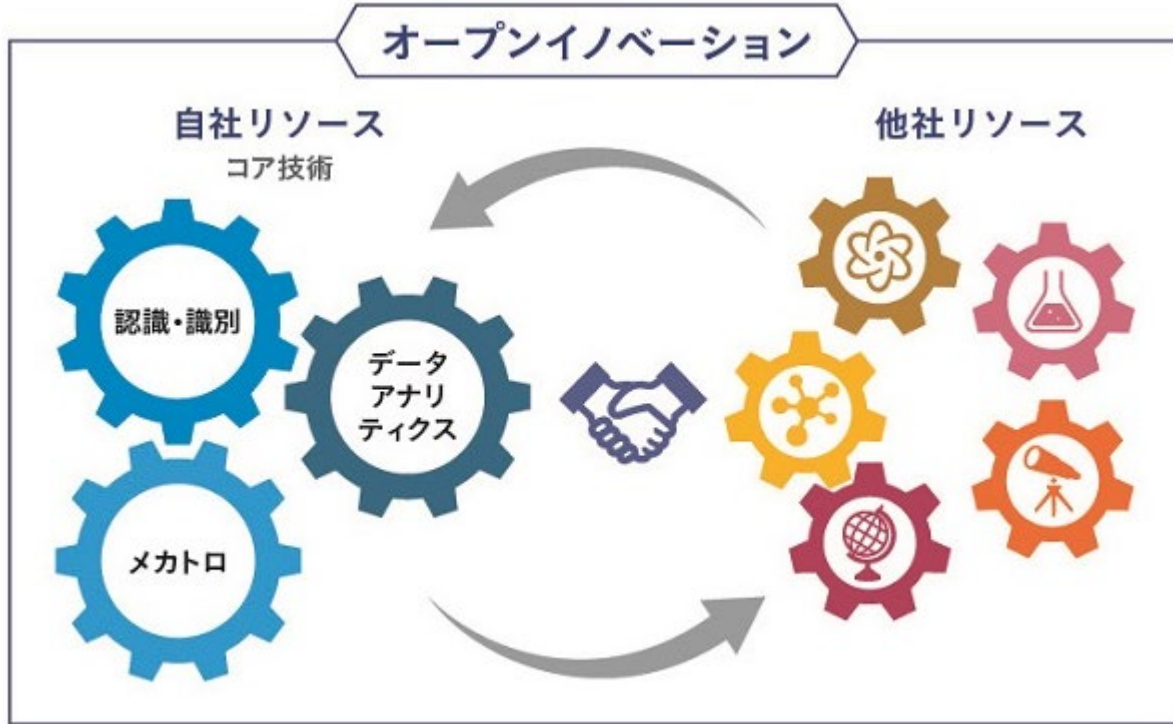
取り付けイメージ



タッチレス操作可能になる
フレームを開発

本フレームは、券売機
「VT-T20」と診療費支払機
「FFH-700」のオプション
として2021年度中に提供を
開始

MBD導入の背景



新コア技術の開発
⇒新しいビジネス展開

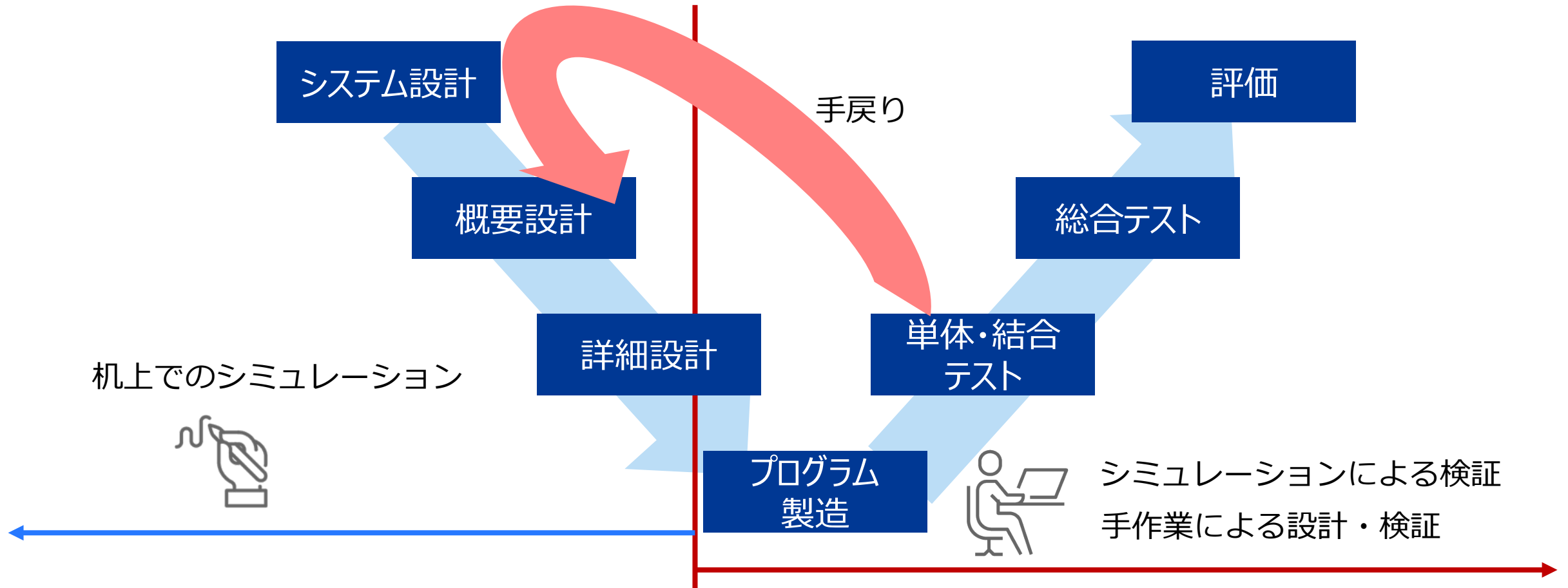


従来業務：既存製品の開発／保守

両立させるには従来業務の生産性向上が不可欠

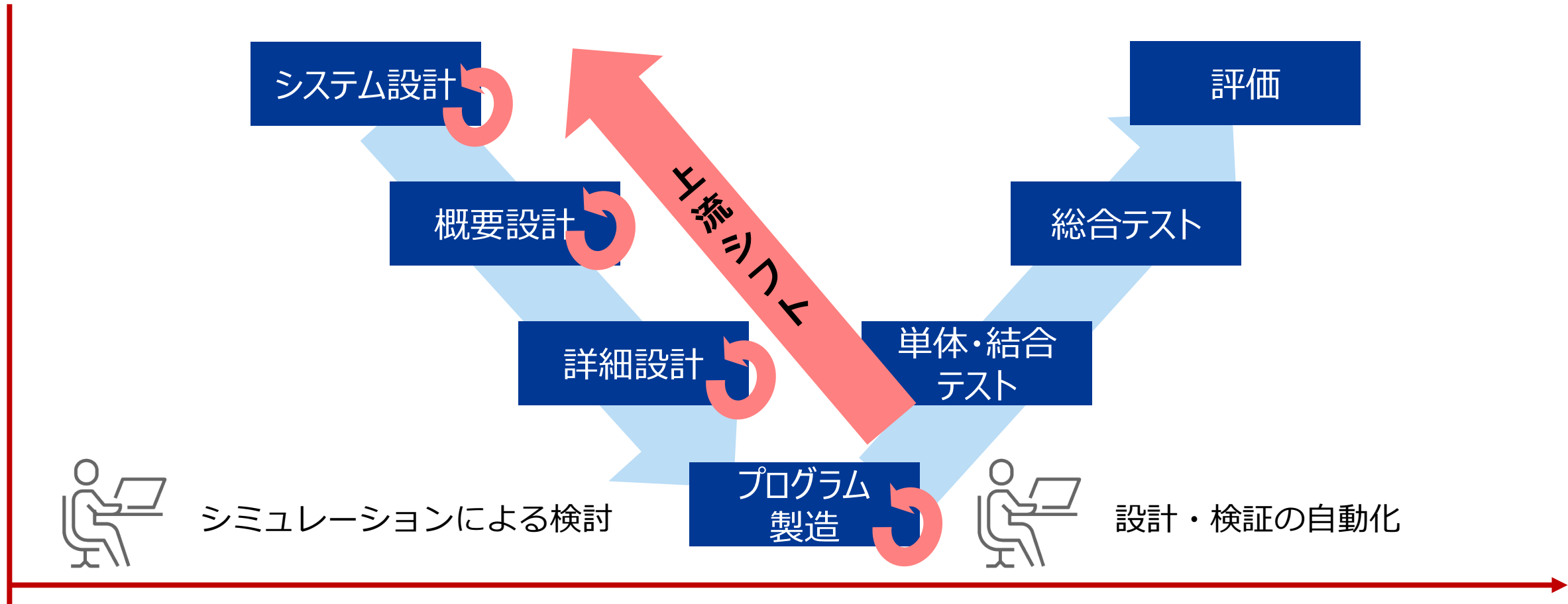
MBD導入の背景 ～ 現状の問題点 ～

- ・仕様書は文章・図で構成されており、シミュレーションは机上の検証
- ・独自システムによりシミュレーションを実施しているが、工程としては中盤
- ・手作業による設計・検証 工数大



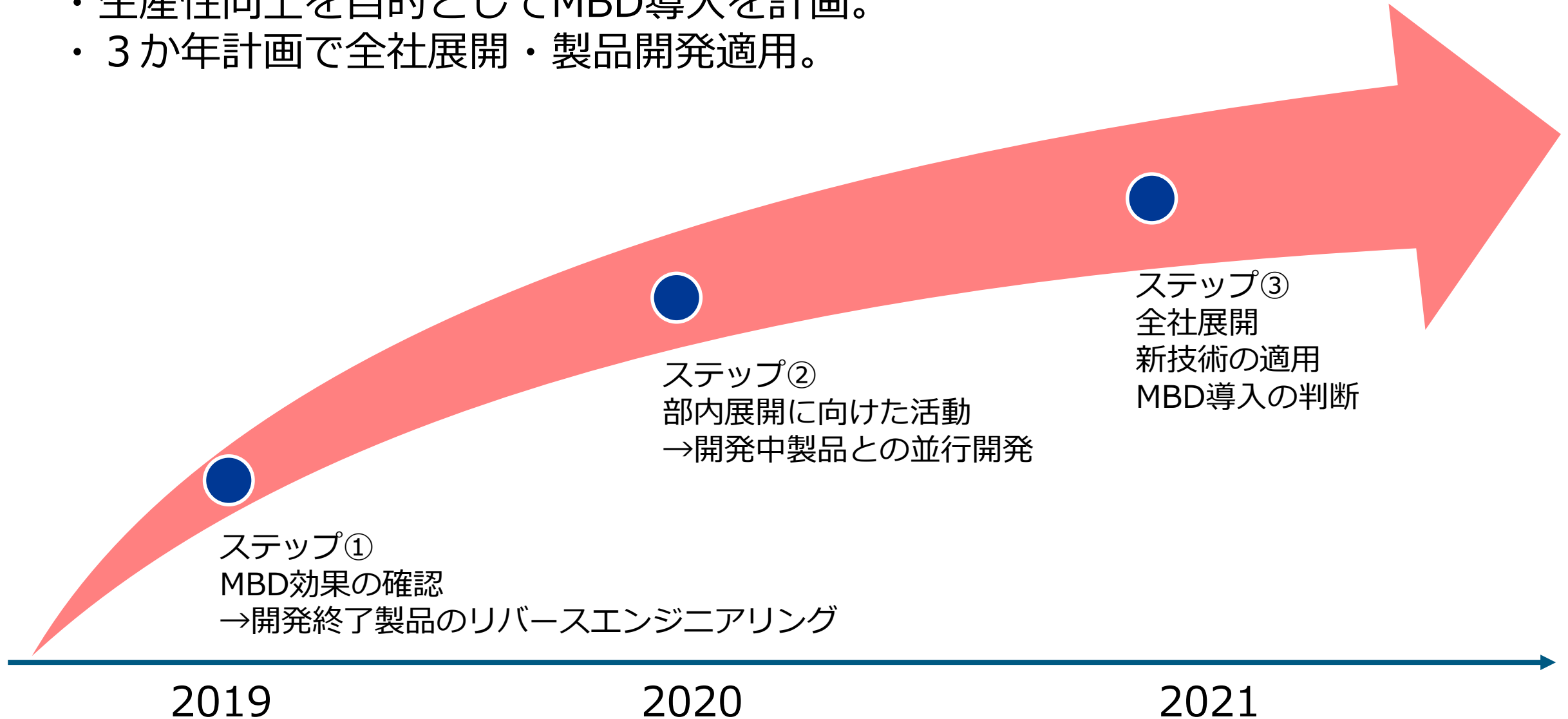
MBD導入の背景 ～ 理想とする姿 ～

- MBDを導入、仕様のモデル化による シミュレーション範囲の拡大
- 上流シフトによる 不備の早期発見・手戻り工数の削減
- コード作成・検証の自動化による 工数削減



MBD導入計画

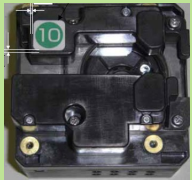
- ・生産性向上を目的としてMBD導入を計画。
- ・3か年計画で全社展開・製品開発適用。



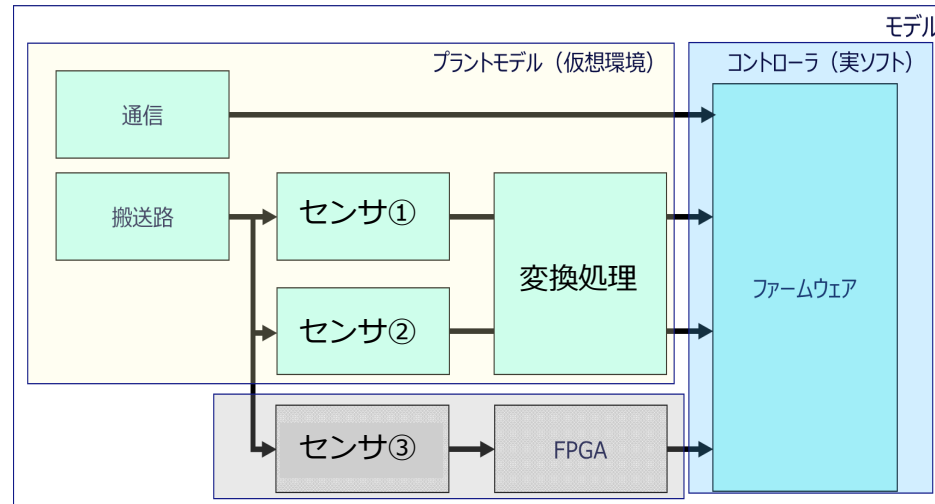
導入初年度の成果

ステップ①として弊社開発の識別装置をターゲットにしてモデル化を実施。
コード生成し、SpeedGoat社のハードウェア上で動作させることを目標とした。

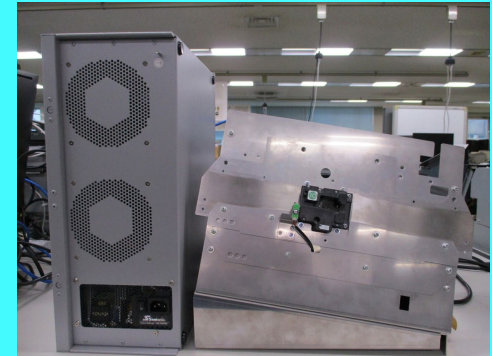
Input



ソースコード



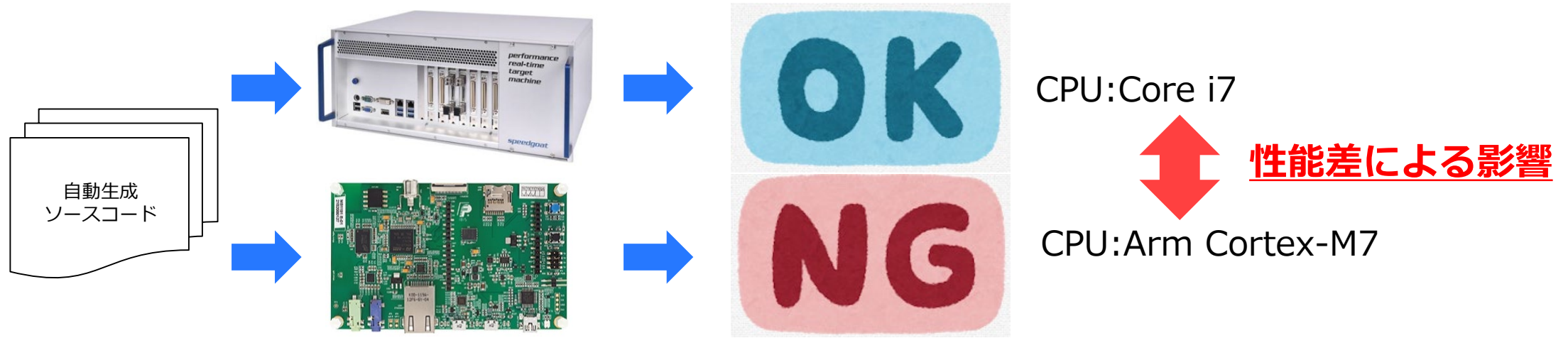
検証



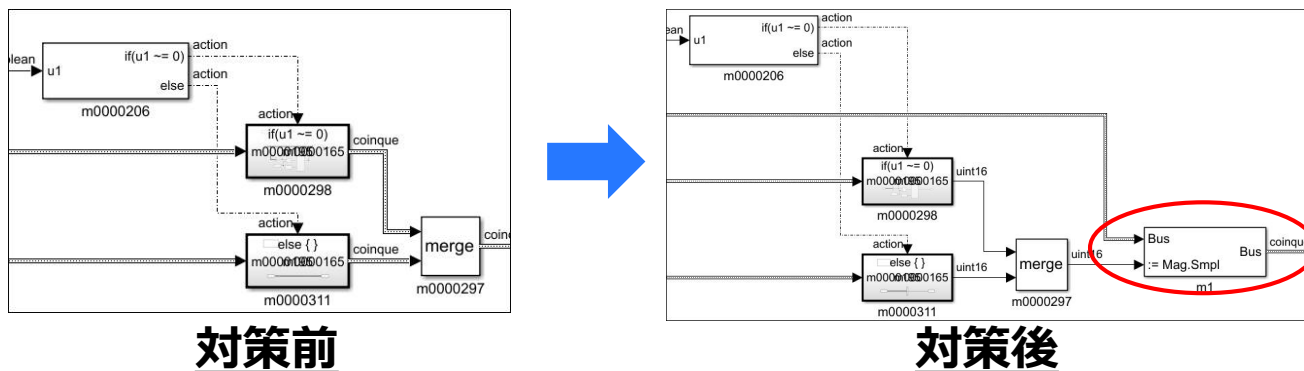
生成コードに問題なく、正常に識別できていることが確認できた。
モデル化工数も少なく、従来開発に比べ工数削減できるという結果も得られた。

導入初年度で分かった課題

- ・ 実機のボード使用時、**処理時間が7倍**に増加。
実運用では耐えられないことが判明。



- ・ 上記問題に関する対策をMathWorks様へ相談。
モデル構造に問題があったことが判明



- ・ 左記対策により**問題は解決**。
- ・ 開発手順、ルール化など
製品適用にむけ**課題は山積み**。

製品適用に向けた活動

- ・ MBD立上げ支援プログラムでは、MathWorks様・弊社と共同で活動。
下記体制で実施。

MathWorks様

Role	Name	Department
技術担当	袁 様	アプリケーションエンジニアリング部
技術担当	柳原 様	アプリケーションエンジニアリング部
技術担当	田中 様	アプリケーションエンジニアリング部
技術担当	鎌谷 様	アプリケーションエンジニアリング部
営業担当	豊倉 様	クロスインダストリ営業部
営業マネージャー	堀川 様	クロスインダストリ営業部

弊社

Role	Name	Role	Name
MBD:Chief Engineer	藤田	製品開発:Chief Engineer	織邊
MBD:Manager	小林	製品開発:設計者	片岡
MBD:設計者	中川	製品開発:設計者	眞殿

製品適用に向けた活動



・ MBD立上げ支援では、MathWorks様に下記の協力をいただき活動実施。

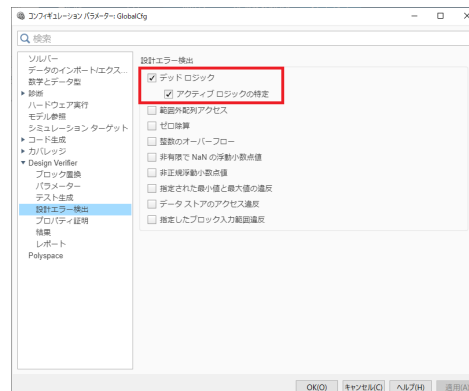
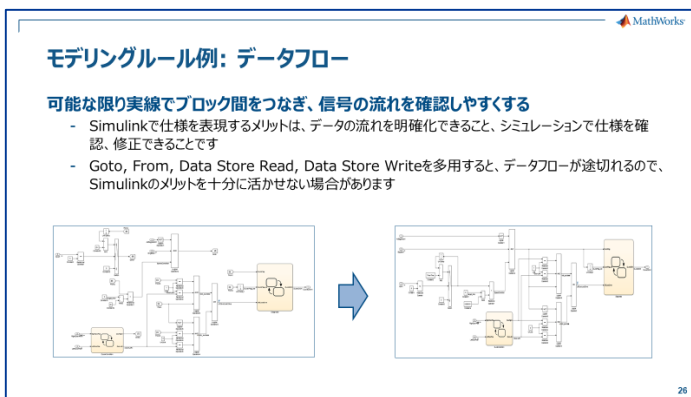
① 定例ミーティングの開催（Teamsによる会議・月1回：2h）



② 活動推進中の不明点に関するメールベースでのサポート（随時）

No.	日付	担当	対象	Priority	相手	質問	回答	再質問No.	Corresponding state
1	2018/06/28	田中	MBD 立上げ支援	大至急	MathWorks	TruthTableでの記載は型変換が必要か？	StateFlow内の言語表現を任意に切り替えることができます。(プロ/ドキュにて変更可)		クローズ

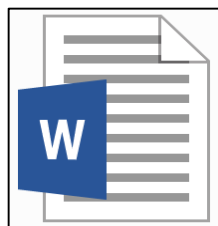
③ MBDワークフロー構築・改善へのアドバイス・参考となる資料とサンプルの提供（随時）



・ 弊社のプロセスと融合させ、開発中の製品に並行してMBDを適用。
製品展開を目標に活動を行い、従来開発とMBDの品質コスト面を比較する。

ソフトウェア要求仕様の整理

- 要求仕様の整理にUSDMを活用（弊社プロセス）
- ※USDM(Universal Specification Describing Manner)



仕様書

半径算出動作	(上位要求番号)	目標テープ速度を計	
		ドラム径が不確か	
		安全に動作しない	
理由			
説明		要求に対する補足などを記述する	
<テープ位置確認>			
	□□□	(仕様番号)	テープ位置を確 A)テープエンド B)それ以外の場
<モーター起動待ち>			
	□□□	(仕様番号)	モーター起動待 [モーター起動待 A)RSM異常検出 <緊急停止処理 B)タイムアップ時
	□□□	(仕様番号)	

理由を書くことで、仕様を明確にでき、認識のずれを排除できる

階層化されることで仕様が漏れにくい



取込

USDMに手を加えることなくモデルに取り込みが可能

USDMの内容をモデルから参照可能

USDM
(Excelで作成)

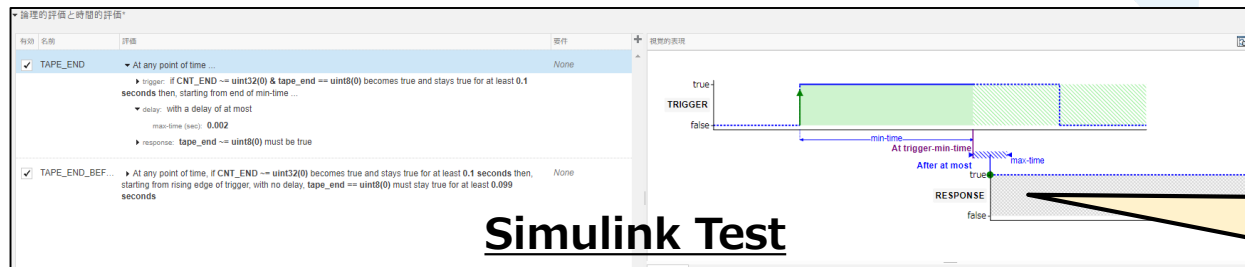
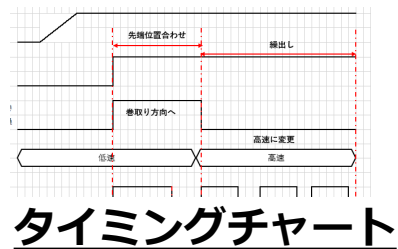
Simulink Requirement
(要件エディタ)

仕様書の不備を検出、モデル上での仕様管理を実現

ソフトウェア要求仕様の可視化と詳細化



・ タイミングチャートによるソフトウェア要求仕様の可視化



Simulink Test使用で
タイミングチャート通りの
テスト作成ができる。

・ 制御構成図・入出力期待値によるソフトウェア要求仕様の詳細化

申請書出動作	(上位要求番号)	目標テープ速度を決定するにあたり、ドラム径の計測(ドラム径が不揃いな場合(電源の断線)、RSM駆動モーター安全にドラム径を計測する。計測後、ドラム径とテープ厚に不揃いがないか、チェックする。
理由		
説明		要件に対する補足などを記述する
<テープ位置確認>		
□□□	(仕様番号)	テープ位置確認
<モーター起動待ち>		
□□□	(仕様番号)	モーター起動待ち

USDM



機能の関係性
データのやり取りが
一目でわかるようになる。

下記項目に関して確認
・ 入出力に過不足がないか。
・ 機能が肥大化していないか。

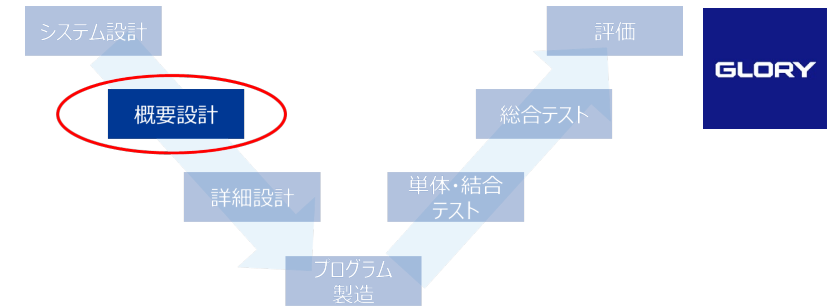
減速タイミング監視

搬送通過 (PRMx1)	前回搬送通過	OSBACK	Notエンドタイプ	テープエンド確定前回値	減速実行
**** : 1	**** : 1	**** : 1	**** : 1	**** : 1	**** : 1
**** : 0	**** : 0	**** : 0	**** : 0	**** : 0	**** : 0
0	0	0	0	0	0
0	1	0	0		
1	1	1			
1	0	1	0		
0	0	0	1	1	0

入出力期待値

機能の可視化、適切な機能分割を実現

アーキテクチャーモデルの作成



モデル階層構造の検討

・ JMAABの階層概念に基づきレイアウトを決定

・ 階層概念

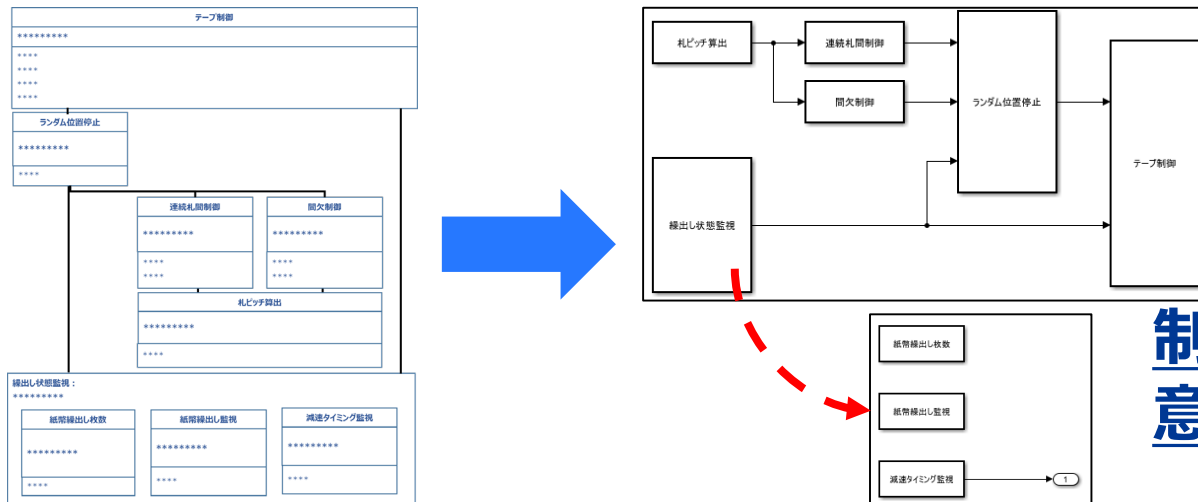
	階層概念	階層の目的
上位階層	機能レイヤ	大きな機能の分割
	スケジュールレイヤ	実行タイミング(サンプリング・順序)の表現
下位階層	サブ機能レイヤ	小さな機能の分割
	制御フローレイヤ	処理手順(入力→判断→出力、など)に沿った分割
	選択レイヤ	アクティブとなるサブシステムを切替えて実行する形式に分割([Merge]で出力を選択)
	データフローレイヤ	分割不可能な一つの計算を行うレイヤ

- ・ 簡易な制御モデル **採用**
機能レイヤとスケジュールレイヤを同一レイヤで表現します。機能＝実行単位となります。
例: 制御モデルが一つのサンプリングのみであり、各機能が実行順序順に配置されている場合
- ・ 複雑な制御モデル タイプ α
スケジュールレイヤを上位に配置します。
コードとの統合は楽になりますが、機能が分断されモデルとしての可読性が下がります。
- ・ 複雑な制御モデル タイプ β
機能レイヤを上位に配置し、個々の機能レイヤの下位にスケジュールレイヤを配置します。

※CONTROL ALGORITHM MODELING GUIDELINES USING MATLAB, Simulink, and Stateflow Version 5.1

スケルトンモデルの作成

・ 制御構成図をInputにSubsystem、Inport、OutPortを配置



制御構成図で処理を明確にしており、意図しない変更を防止できる。

制御ロジックの作成 ~ モデル設計 ~

- Simulink Requirementを使用し、詳細モデルを設計



Simulink Requirement (要件エディタ)

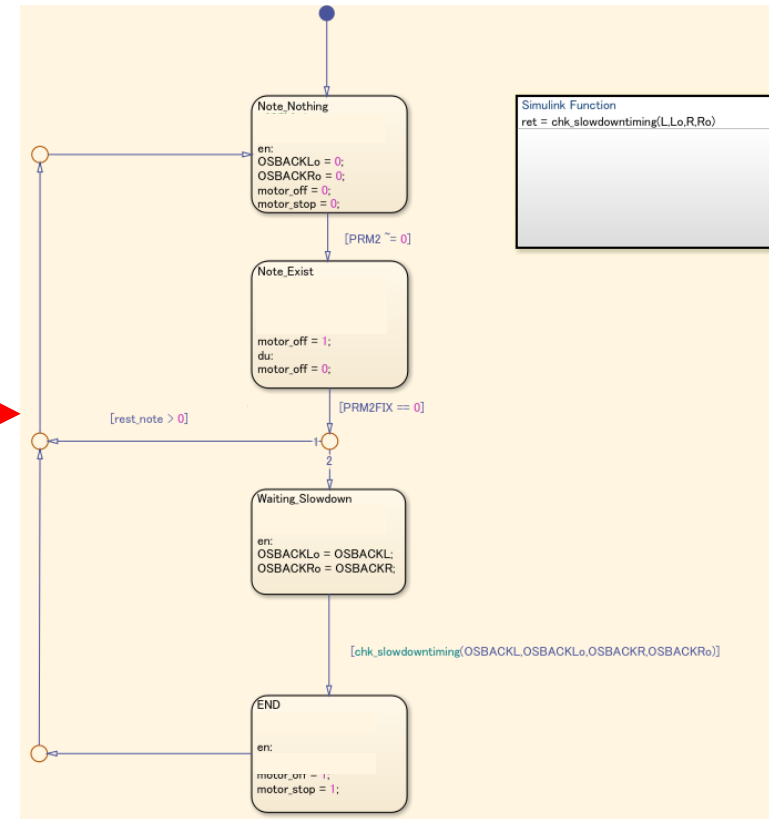
仕様を確認しながら設計が可能

減速タイミング監視

搬送通過 (PRMx1)	前回搬送通過	OSBACK	Notエンドタイム> 100ms	テープエンド確定前回値	減速実行
**** : 1	**** : 1	**** : 1	**** : 1		
**** : 0	**** : 0	**** : 0	**** : 0		
0	0	0	0		
				1	1
0	0	0	1	1	0

入出力期待値

機能詳細はこちらで確認

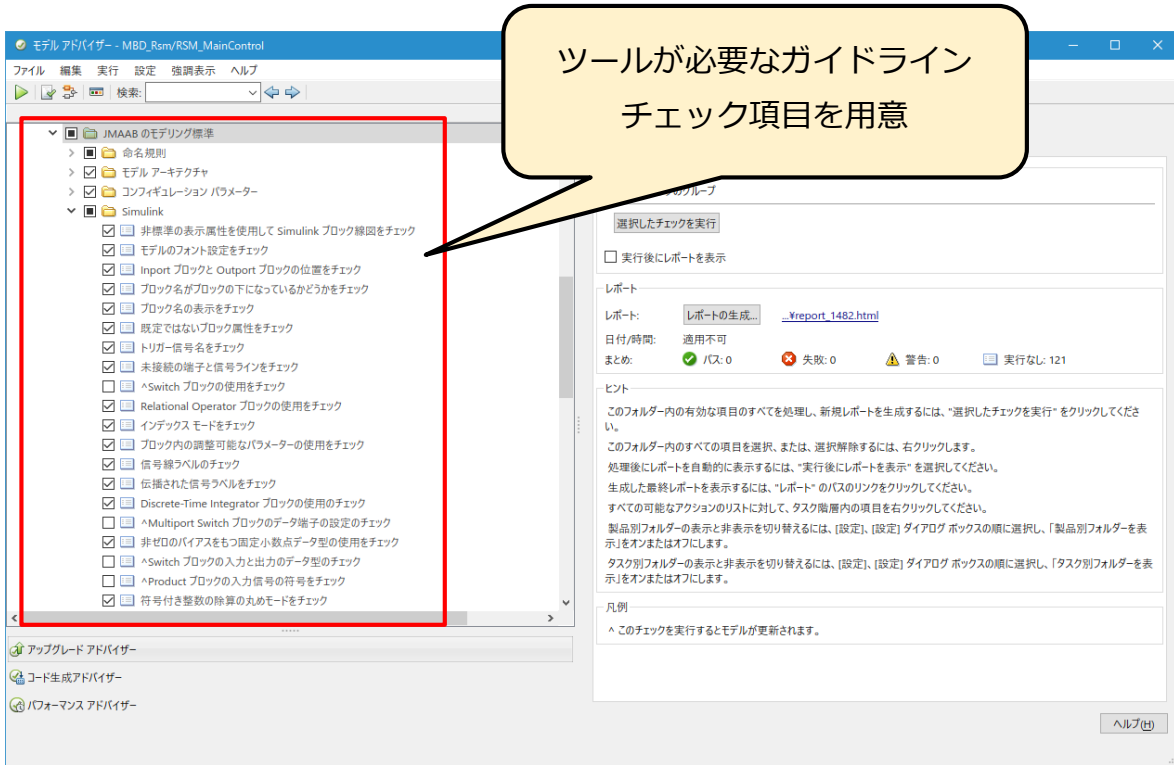


仕様と機能をリンク、実装ミス防止に貢献

制御ロジックの作成 ~ ガイドラインチェック ~

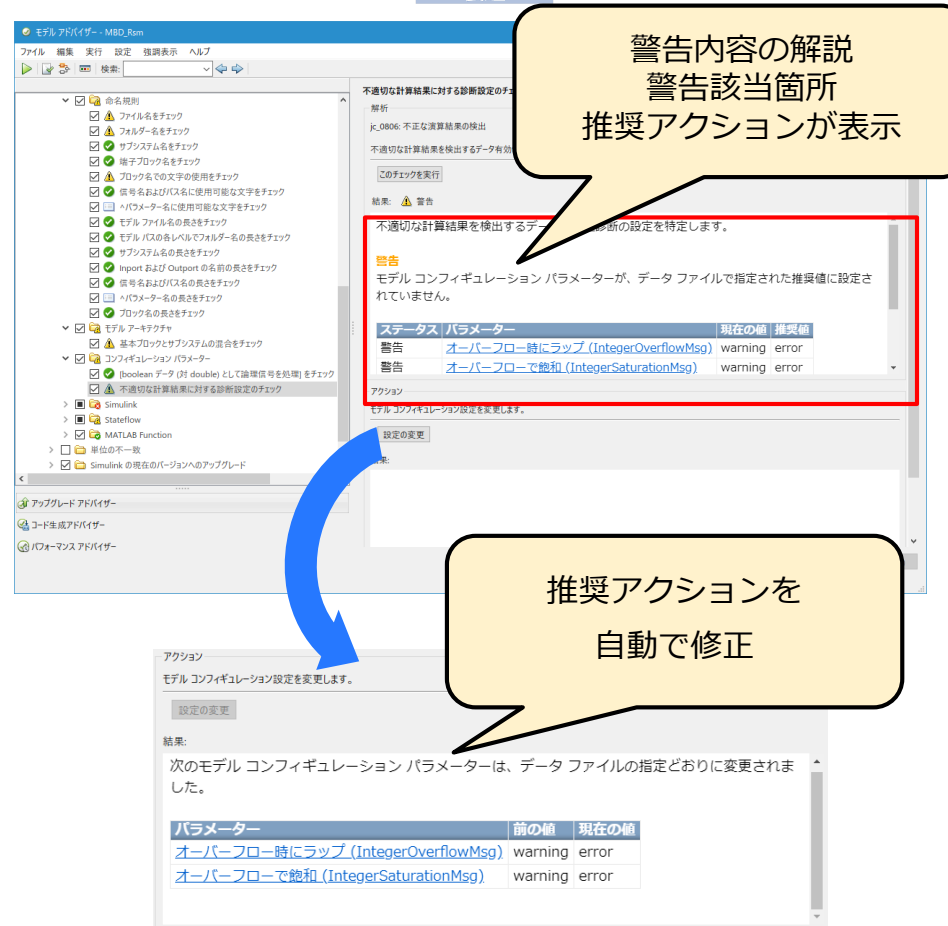


- Simulink Checkを使用し、設計したモデルに対しガイドラインチェックを実施



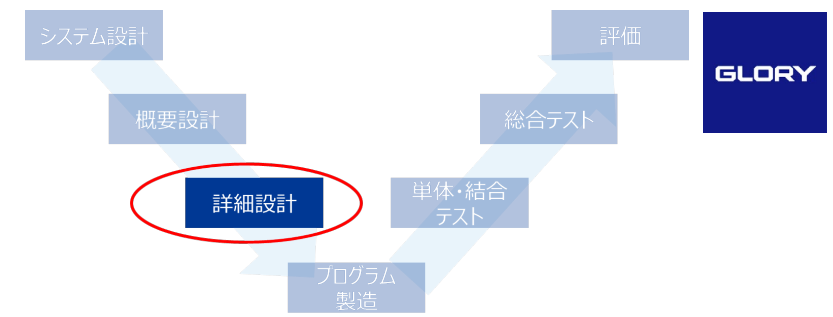
JMAABルールを採用

実行

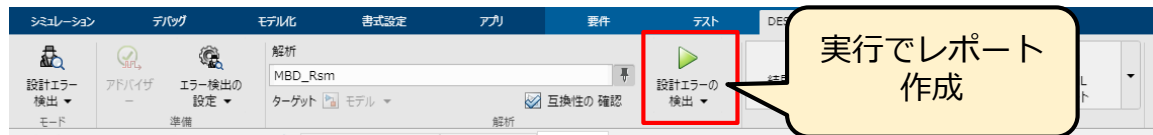


潜在的なミスを事前に取り除くことが可能

制御ロジックの作成 ~ 設計エラー検出 ~



- Simulink Design Verifierを使用し、設計したモデルに設計エラーがないかをチェック。
 ※設計エラー：オーバーフロー、デッドロジック、ゼロ除算等



Simulink Design Verifier

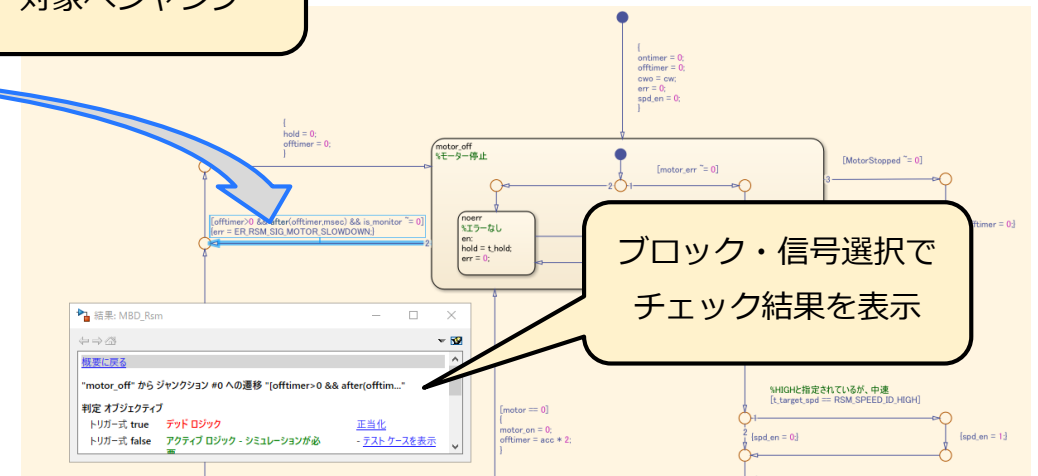
デッドロジック

Simulink Design Verifier により、これらの判定と条件の結果は発生し得ず、モデルにおいてデッドロジックであることが判明しました。モデルのデッドロジックは、パラメーター コンフィギュレーションや、入力に指定されている最小制約および最大制約の二次的影響である場合もあります。

#	タイプ	モデル項目	説明	解析時間 (秒)	テストケース
646	判定	"motor_off" からジャンクション #0 への遷移 "[offtimer>0 && after(offtim...]"	トリガー式 true	56	n/a
652	条件	"motor_off" からジャンクション #0 への遷移 "[offtimer>0 && after(offtim...]"	"is_monitor ~= 0" true	56	n/a
675	判定	"motor_on" からジャンクション #1 への遷移 "[ontimer>0 && after(ontimer...]"	トリガー式 true	56	n/a
681	条件	"motor_on" からジャンクション #1 への遷移 "[ontimer>0 && after(ontimer...]"	"is_monitor ~= 0" true	56	n/a
900	判定	RSM_MainControl/winding/slowdown/note_select/Saturation	入力 > 下限 false	56	n/a

チェックレポート

ハイパーリンクで対象へジャンプ



対象モデル

複雑なロジックも自動でエラー検出が可能

モデルの検証

システム設計

概要設計

詳細設計

プログラム製造

単体・結合テスト

総合テスト

評価

GLORY

- Simulink Testを使用し、設計時に検証コンテンツを作成

要件 (Simulink Requirement) とリンク

線出し札ピッチと札間の測定

カスタムID: 線出し札ピッチと札間の測定
概要: 線出し札ピッチと札間の測定

説明 概要
線出し札ピッチと札間の測定

紙幣の先端が搬送通過センサ(マスク溶PRX*2)に到達したタイミング(0)以降(有期)で、OSBACKを記録し、紙幣の後端が搬送通過センサ(マスク溶PRX*2)を抜けたタイミング(0)以降(無期)で、OSBACKの差分にて札長情報を取得する。

キーワード

ファイル: D:\gitproject\...

タグ

説明

要件*

線出し札ピッチと札間の測定: 線出し札ピッチと札間の測定 (V20#120)

テスト対象システム*

モデル: MBD_Rsm

テストハーネス*

ハーネス: MBD_Rsm_cal_notepitch_Test

シミュレーション設定のオーバーライド

パラメーターのオーバーライド*

コールバック

入力*

シミュレーション出力*

コンフィギュレーション設定のオーバーライド

ベースライン基準*

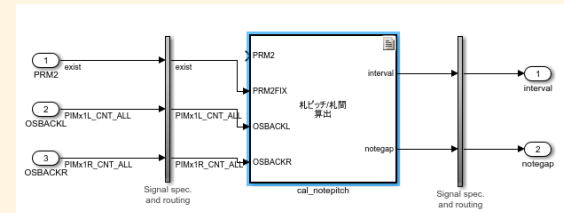
反復

論理的評価と時間的評価

カスタム基準

カバレッジの設定*

テスト対象とリンク



入力・基準は
入出力期待値に合わせて
Excelで設計

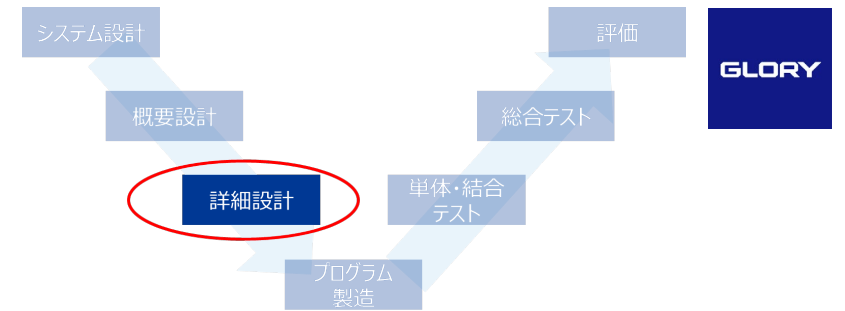
time	PRM2	OSBACKL	OSBACKR	Parameter: Value:	time	OSBACKL
0	0	0	0	RSM_OSBACKRSM_0	0	0
0.1	1	0	0		0.1	0
0.2	0	80	72		0.2	0
0.3	1	200	180		0.3	0
0.4	0	280	252		0.4	0
0.5	1	400	360		0.5	0.01
0.6	0	480	432		0.6	0.01
0.7	1	600	540		0.7	0.01
1	0	700	630		1	0.01
1.5	0	700	637		1.5	0.01
2	1	797	717		2	0.01
2.1	0	779.3	697		2.1	0.01
2.2	1	843.3	937		2.2	0.01
2.3	0	951.3	1057		2.3	0.01
2.4	1	1023.3	1137		2.4	0.01
2.5	0	1176.3	1307		2.5	0.02
2.6	1	1248.3	1387		2.6	0.02
2.7	0	1248.3	1387		2.7	0.02
3	0	1254.6	1394		3	0.02

減速タイミング監視

搬送通過 (PRMx1)	前回搬送通過	OSBACK	NotEndTime > 30ms	テープエンド確定前回値	減速実行
**** : 1	**** : 1	**** : 1	**** : 1	**** : 1	**** : 1
**** : 0	**** : 0	**** : 0	**** : 0	**** : 0	**** : 0
0	0	0	0	0	0
0	1	0	0	0	0
1	1	1	0	0	1
1	0	1	0	1	1
0	0	0	1	1	0

関連資料の一元管理により作業効率化を実現

実装・検証の進捗管理



- Simulink Requirementを使用することで、実装・検証の進捗管理を実施。

要件エディタ

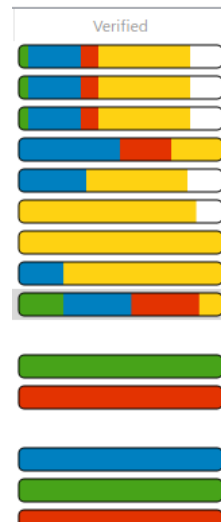
更新が完了しました。変更はありません。

Index	Summary	Implem	Verified
V20*			
イン...	GLR-200_メカ制御仕様書_(RSM制御...		
1	RSM/ISM600制御		
	RSM/ISM600部概要		
	テープ制御		
	監視制御		
	テープ位置監視		
	ドラム径検知制御		
	紙幣滞留監視		
	位置合せ処理		
	位置合せ		
	巻取り処理		
	繰出し処理		
	繰出しモード		
	センサマスク		
	繰出し札間制御		
	繰出し札ピッチと札間の測定		
	駆動モーターの減速と加速		
	シークンス		
	前処理		
	先端位置合せ動作		
	繰出し		
	位置合せ動作		
	後処理		
	残留情報		
	リセット処理		
	指定距離移動処理		
	緊急停止処理		
	バックアップデータ		

要件に対する実装の実績

要件に対する検証の実績

- 検証に関しては詳細に結果を表示



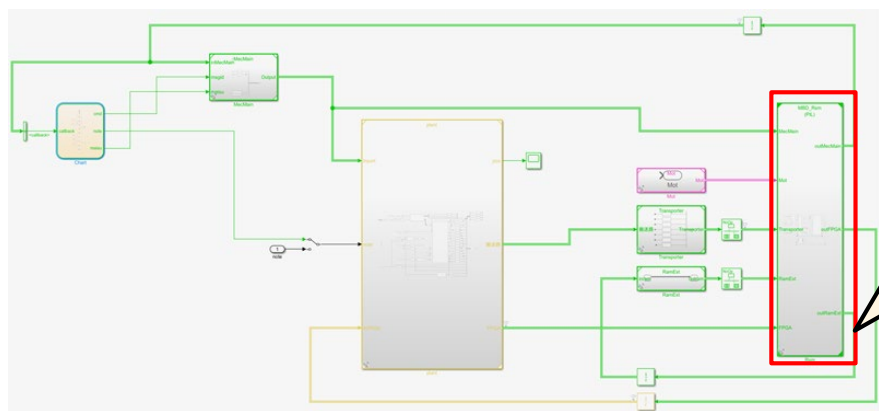
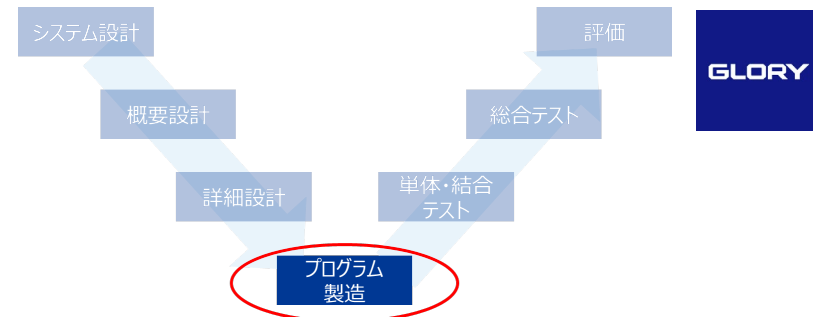
- : 検証OK
- : 検証NG
- : 検証未実施
- : 検査不要

検査が完了したかどうかの判断が一目瞭然。

Simulink Requirementにて管理業務を円滑に遂行

コード生成・実装

- 生成コードを汎用ボードに実装。
PILSにてパフォーマンス測定を実施。
※PILS (Processor In The Loop Simulation)



PILに変更

ブロックパラメータ: Rsm

Model Reference
指定したモデルを参照します。

メイン インスタンスパラメータ

モデル名: 参照... モデルを開く

シミュレーションモード: プロセッサインザループ (PIL)

コードインターフェイス: モデル参照

対象の汎用ボードに
ターゲットを変更

ハードウェアボード:

コード生成のシステムターゲットファイル: [ert.tlc](#)

シミュレーション実施

2. Profiled Sections of Code

Section	Maximum Execution Time in ns	Average Execution Time in ns	Maximum Self Time in ns	Average Self Time in ns	Calls
MBD_Rsm_initialize	954	954	954	954	1
MBD_Rsm_Init	4935	4935	4935	4935	1
[+] MBD_Rsm [0.001 0]	71125	43468	46509	30425	25001

周期処理の処理時間を確認できる

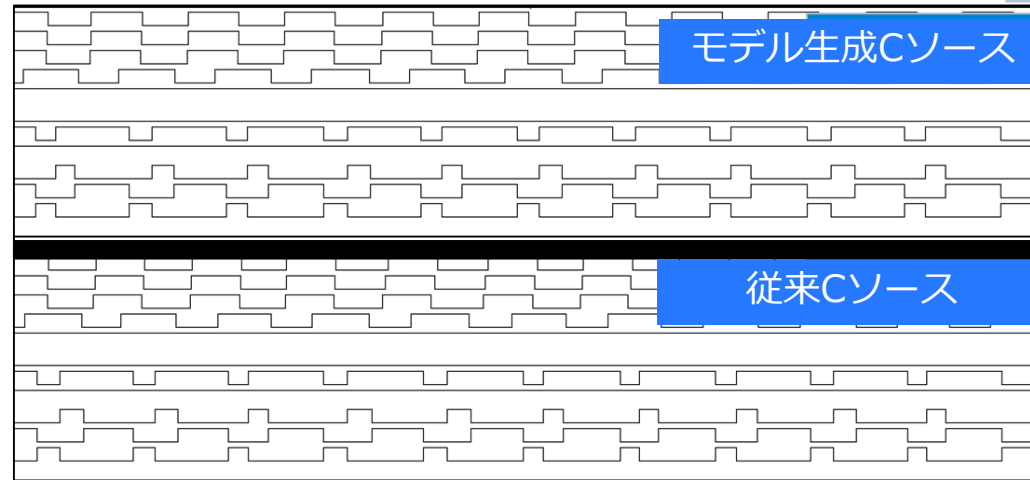
コンポーネント単位での検証が可能、検証の前倒しを実現

製品での検証

- 最終確認として製品で検証



ログチャート出力



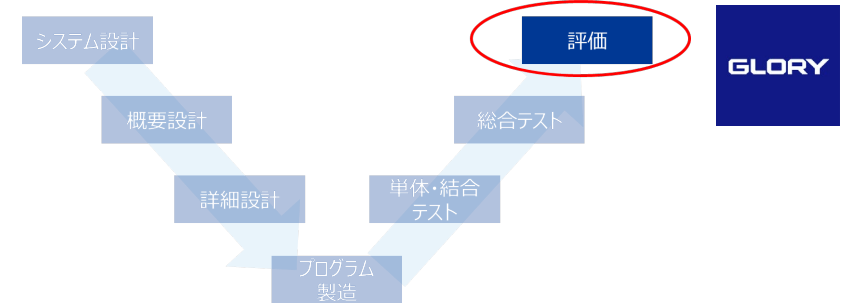
出力一致

検証として下記結果が得られた。

- ログチャートの出力結果が一致。（周期遅れ等も発生無し）
- 処理負荷は減少（従来比数%Down）
- ROM/RAM使用量は増加。（従来比10数%）搭載には問題無しレベル。

MBD立上げ支援当初の目標であった、

MBDで作成したコードが製品適用できることを確認した。



効果の確認

- ・従来開発との比較結果

- ①上流工程での仕様不備発見

USDMを活用した仕様整理の技術により、**仕様書の不備を10件検出**

- ②ソフトウェアにおけるバグの早期発見

Simulink Requirement使用による仕様 ⇔ モデルの関連付け

Simulink Test使用によるソフトウェア検証の前倒し

⇒ 従来開発では未発見の**不具合7件を検出**

- ③開発工数

Embedded Coder、Simulink Check、Simulink Design Verifier使用で
コード作成、ガイドラインチェック、設計エラー検出を自動化

⇒ 従来開発に対し**工数36%削減を実現**

MBD導入により、品質コスト面での効果を確認できた。

まとめ

MBD導入の効果は確認できましたが、課題が残っています。

- ・ 課題

弊社内でのMBD認知度が低い。

→MBD説明会、ハンズオンセミナーの開催、社員教育の実施。

MBD技術者が足りていない。

→人材育成のカリキュラムを整備し、即戦力となる教育システムを作成。

開発工程、設計手法についてのルールが制定されていない。

→標準化に向けたガイドラインの制定

まとめ



MBD立上げ支援での振り返りとしてMathWorks様からご提案をいただきました。

製品/機能紹介やディスカッション等でご支援できると考える項目		
No.	課題	対策案
5	シナリオを作成してテストを作成するか、SLDVIに任せて自動でテストを作成するか。 今回はまず仕様書に従いシナリオを作成し、カバレッジ未達ならSLDVIにさせるという形をとった。	フェーズに分けた運用の仕方。 単体テスト→SLDV 統合テスト→シナリオ作成+SLDV
7	処理負荷に合わせて、subsystemをInlineで作ってしまうべきか。 →開数のオーバーヘッドにより処理負荷は少し上がっているようなので、全てInlineで作れば処理負荷は下がるが、それ以外の何かあったらどうなるか。	適用するシステムに分ける。 そこは↑と同じ考えだが、ここではサブシステムの最小単位を決めておく。 →単体テストの最小単位にもなることなので。
8	そもそもB2Bは必要なのか。 Gloryというモデル設計をしたことがない人であればモデルとソフトで差分が出る可能性がある時点で使いたくないと思われそう。	個人的には単なる一致確認なら不要と考える。
9	上記と合わせて、割込みや処理負荷の影響でモデルでのシミュレーションと違う結果が出るのであればまだいいが、SILS、PILSはそういうわけではないですかね？ →上でやりたことはSimulink Realtimeを使ってボード上で動作させ確認するイメージ	SIL、PILをする場面を定義しておく
12	メカが先にできている状態でソフトが動作できるか不安 →メカ納期的に満足して動作させられるソフトを提供できるか。	メカ屋さんや単体でメカを動作させる仕組みづくりをする。 →メカ屋さん側でソフトと動作確認してもらい、ソフト側は開示しない。
13	ハードの情報が決まっただけから進んでいる感じがするが、実情はそういう風にはないのか、変化に対応しきれないか不安	変化に柔軟に対応できる仕組み or 変化しないように上流で検討できる仕組み。
17	仕様をFixしたものを想定したところで進んでいる印象があった。 Gloryの開発としては仕様が日々変化している中で、それがあった活動が必要ではないかと感じました。	上流でのMBD運用

- 物理モデリング関連の製品/機能紹介
 - システム全体のモデル化
 - モデルからコード生成
- モデル検証/コード生成のディスカッション
 - テストケース自動生成によるカバレッジの向上
 - B2Bテストの進め方/必要性
 - 生成コードの効率化

MW担当	AE (アリエールス)
提供形態	無償
対応時期	随時

コンサルティングにてご支援できると考える項目		
No.	課題	対策案
1	モデルを作成する準備段階として、開発環境、ハードウェアの設計について整理する点ではおこなっていただくまでであったのでは？ 使用するマイコン、RTOS等、意識しないなら意識させたいレベルまでおこなっていただく。	設計書等のInput確認を義務とする。 制約構成図や入出力期待値のよみに必要なものを上げる。
2	制約構成図に基づいて機能分割を行うべきでない、制約構成図をいじるともとの実装したときに分割されずにつながりがおこしやすくなり、再度制約構成図を再修正などの作業が発生した。 →入力出力の接続を再確認できない、出力が足りず制約が超過する。	JSDM等の要求の整理を行う。
3	機能名と出力があてはまらない、実装する過程で効率化の手間や出力を増やしている。 →分割しきれない。	アバウトなドキュメントの基礎を決める。 →個人での作業にならないようにする。
4	極々の機能でのテストは問題なくても場合時、実機で問題が発生することがある。 →単体テストではデータが矛盾している、あるいはユーザーでテストしていた。外部からのデータはある程度排除できる。そこから該当の機能に到着するまでのデータの变化をどうまで追うことができるか。また追えなくても効率のテストできるか。	入力データの関連を示す資料の作成 →他のデータの依存関係、帰反、連鎖などを示すもの 上記に変わるものとしてはフロントモデル、メカシス。
6	基本の設定をどうするか。 いったんは作成したが、それが良いのか判断が自分ではわからない。	何種類かの設定パターンを用意し、どのシステムで採用するかあらかじめ決めておく。 処理負荷削減、RAM容量削減など。
10	外部RAMの更新や、開数コールをどう表現するべきか迷う。 更新データをOutputにする。 →トランザクションを無視しないといけない →データが溜まっていくが、更新タイミングを意味するものも必要 →開数コールの場合も同様 →開数コールの使用すると、上記問題を解決するために、DataStoreMemoryやCoilerなどを使用すると、データのつながりがもたずシステムで途切れる、中を見ない何をやっているかわからない →開数コールをCoilerで使ったほうが実装時に有利になるが、モデルとしてはわかりにくく、3Dビューを少しづつ変更。	ケースバイケース、プロジェクトによって、会社方針として必ずすべきか。 DSMを使えば実装は楽になるが弊害も多いので →製品運用時には使わない。検討段階では使用する等。 シミュレーションモデル、実装モデルを分ける。 個人的にはこれはしたくない。

- コンサルティング
 - MBDトライアル
 - ガイドライン作成

* サービス内容は別途ご紹介させていただきます。

MW担当	コンサルティング
提供形態	有償
対応時期	3月以降

* 日程等、詳細につきましては別途お打ち合わせにてご相談させていただきます。

コンサルティングも含め、MathWorks様協力の元、今後もMBD推進をしていきたいと思っております。

最後に、MBD立上げ支援の活動として協力いただきました
袁様、柳原様、田中様、鎌谷様、豊倉様、堀川様に厚く御礼申し上げます。

The logo consists of the word "GLORY" in a bold, white, sans-serif font, centered within a solid dark blue square.

Confidence Enabled

人と社会の「新たな信頼」の創造へ

