

パッケージ開発プロセスの プロセス改善による品質向上

～タイムボックス制約付きインクリメンタル開発～

株式会社富士通マーケティング
ソリューション事業本部 GLOVIA SUMMIT事業部
SUMMITソリューション部

松浦 豪一

注意事項



- GLOVIAは日本およびその他の国における富士通株式会社の登録商標または商標です。
- Microsoft、Windows、Windows Vista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。
- Word、Excel、Internet Explorerは米国Microsoft Corporationの製品です。
- その他、会社名、製品名、名称等の固有名詞は各社の登録商標または商標です。
- 本資料に記載されているシステム名、製品名称等には、必ずしも商標表示を付記していません。
- 本資料の著作権は富士通マーケティングにあります。

1. プロセスが定着している理由



■ ダメなものはダメ

- 定期的なミーティングで必ずチェック
- 相手が分かっている理由を説明する
- 誤りは必ず相手に修正させる

■ 良いものは良い

- 結果を確認して良いものは褒める
- 別の部署の人にも言う(漏れ伝わる)

■ プロセス改善の結果は数値で検証する

- 効果を実感できなければ続かない

2. 私の考える開発プロセスとは



「品質が良いソフトウェアを効率的に開発できる」
という事象を発生させるためにある。



「開発プロセスを**守った**から品質が**保証される**」ではなく、
「品質保証するためには**成果物を総合的に評価する**」
しかない。

品質に問題
ありません。

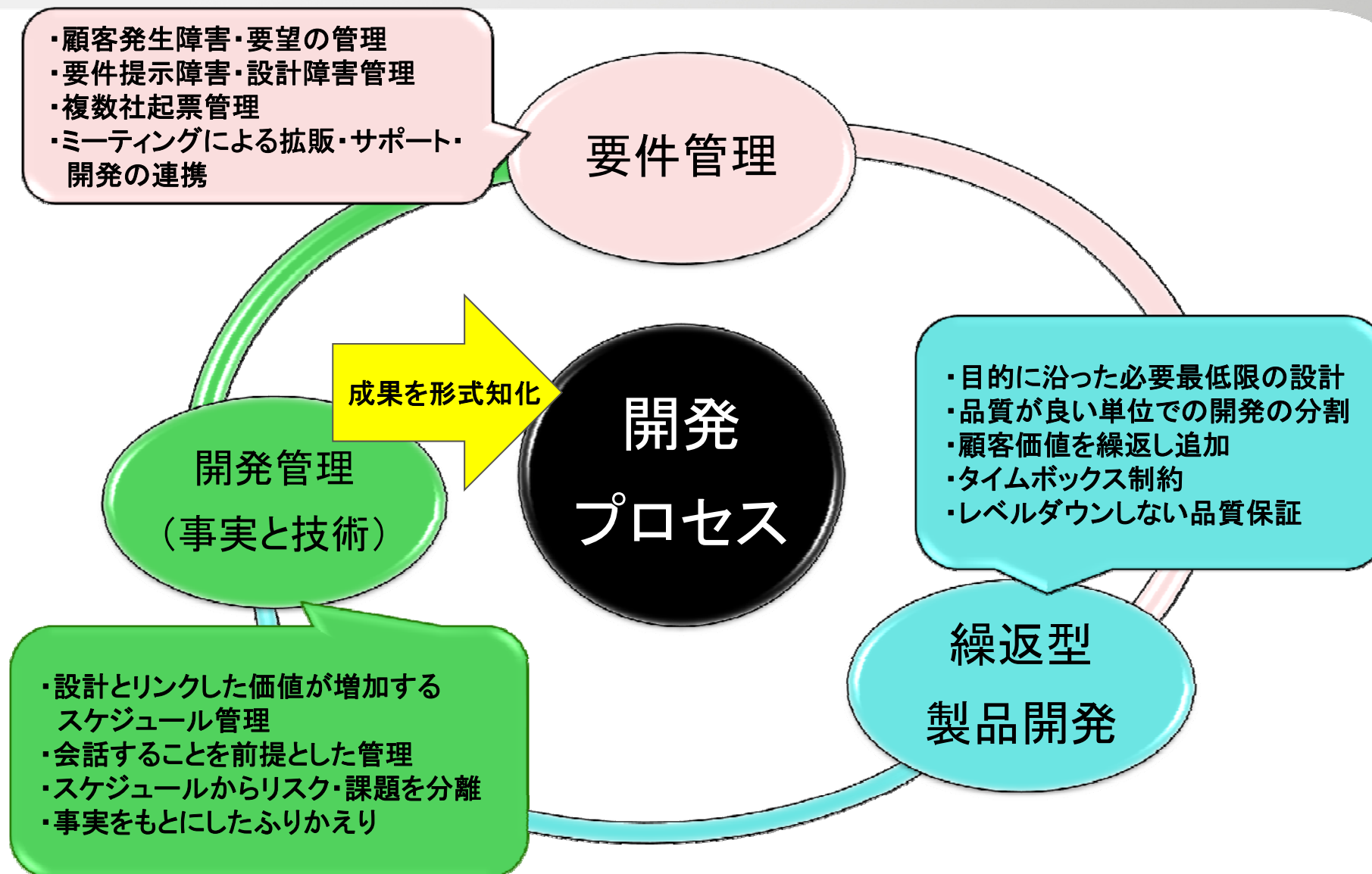


テストデータ作成
に時間かかりました。

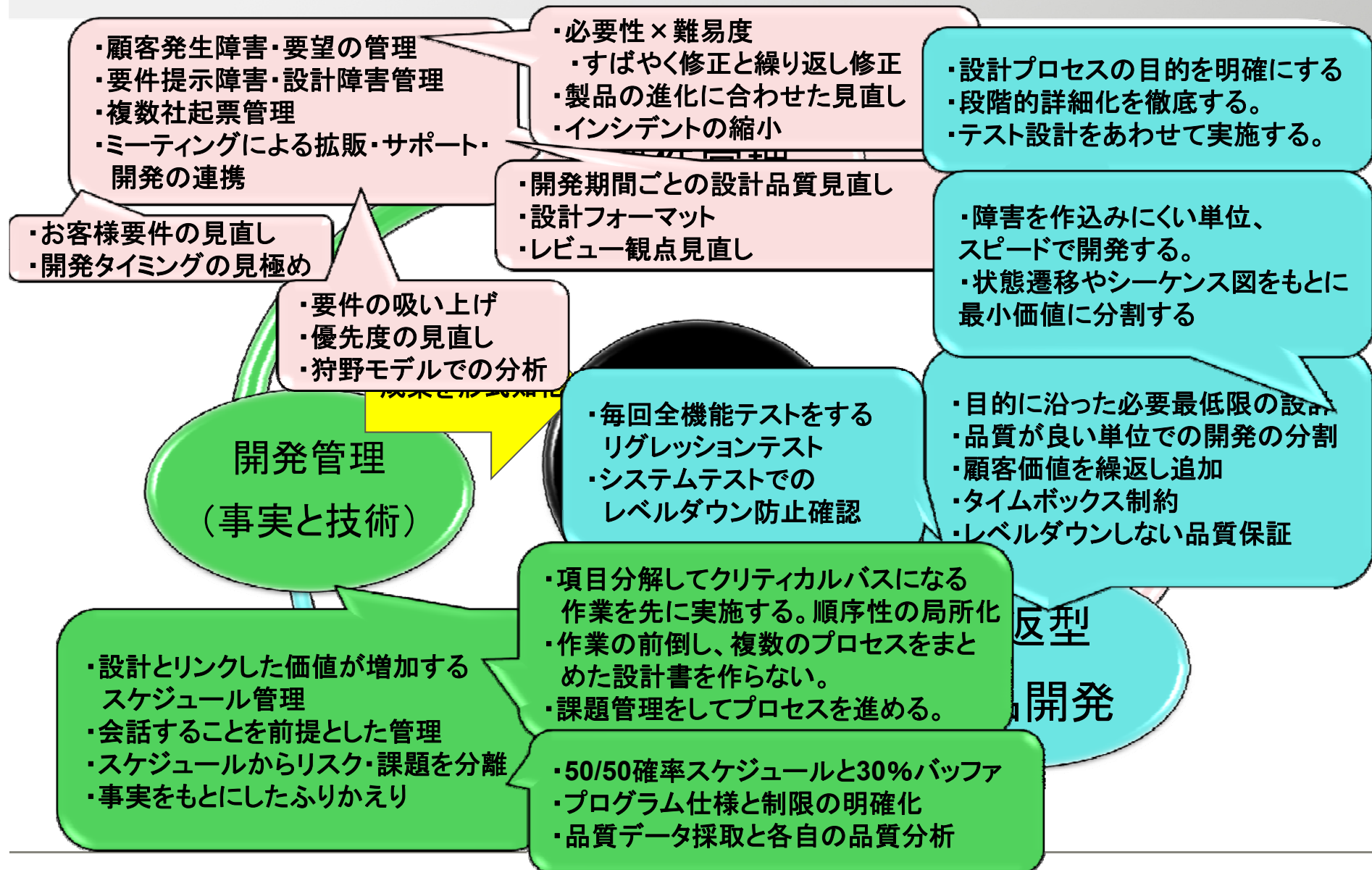
ゆえに、

「品質が保証されている」と「効率良く品質保証できた」
という2つの事象をふりかえる必要がある。

3. 実践している開発プロセス



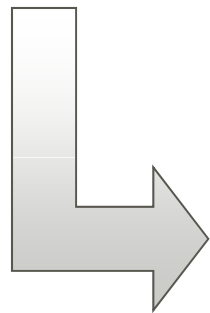
3. 実践している開発プロセス



4. プロセス改善のポイント

背景：リードタイム短縮目的のラウンド開発での失敗
問題点

- 予定期間内に完成しない、見込みが分からない。
- 提供した修正がレベルダウンする。
- 生産性が悪く、成果がすくない。障害対応ばかり

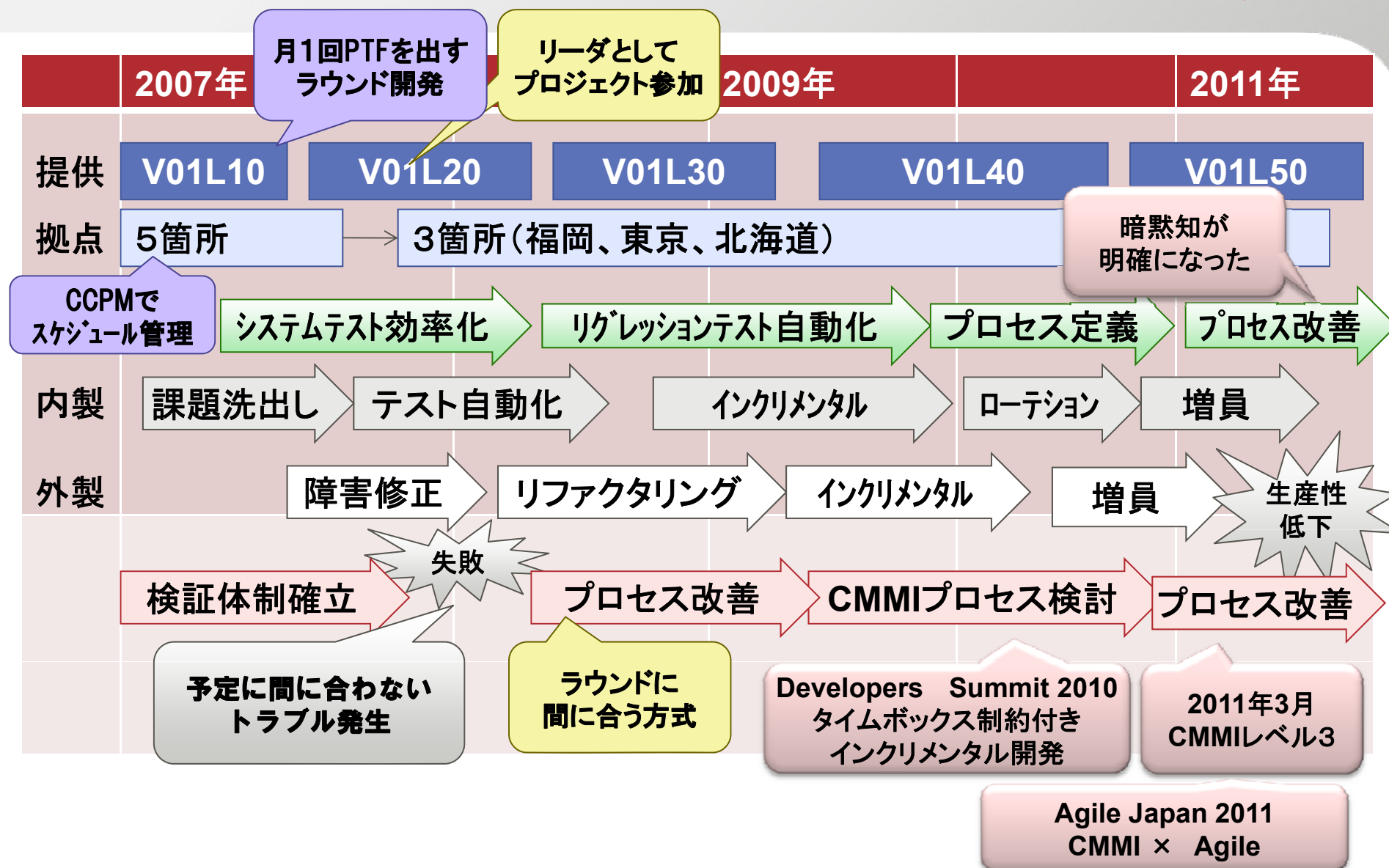


- 品質保証する手段を用意する。
- 障害を作り込みにくい単位で開発する。
- プロセスの目的、価値を見直す。

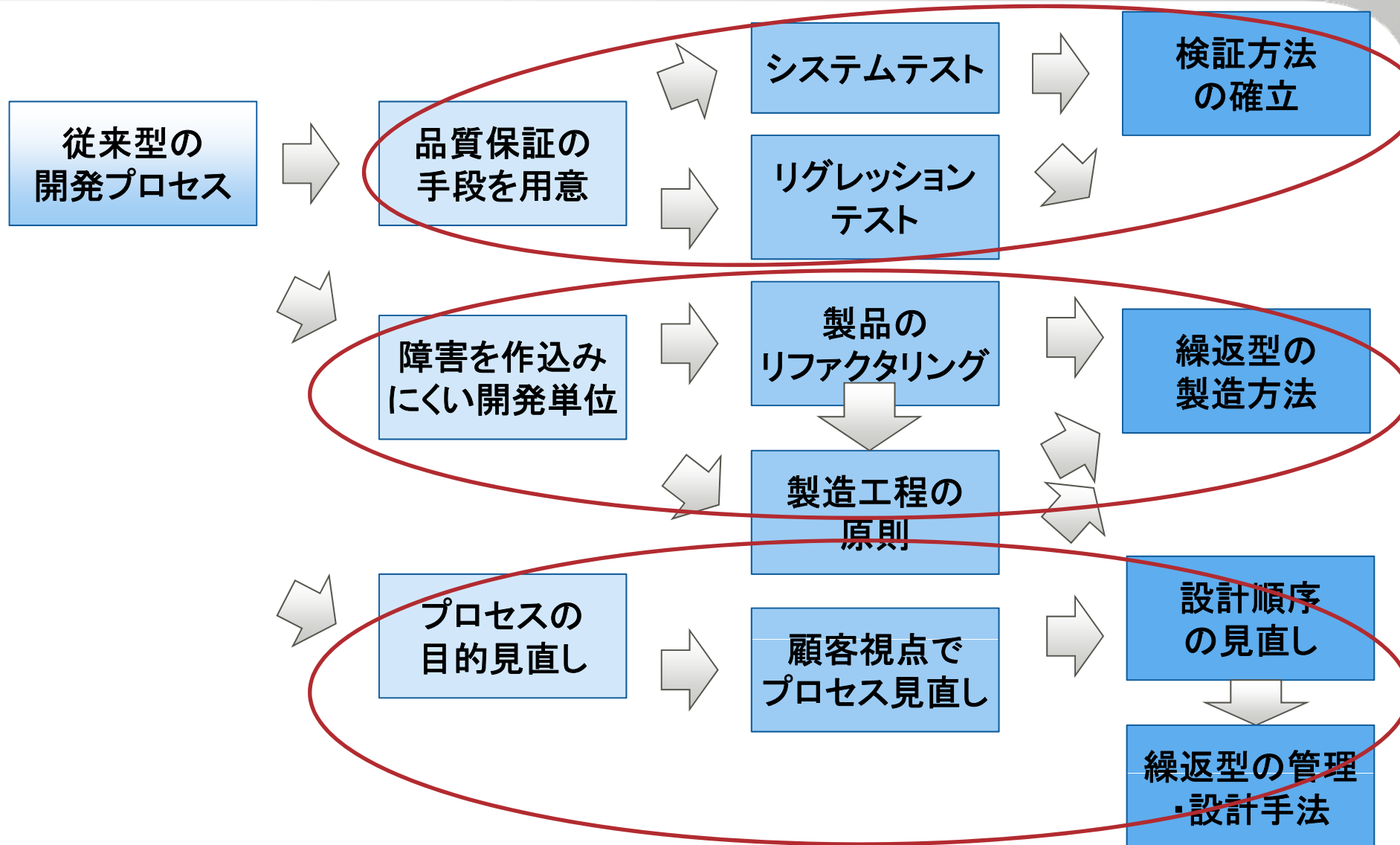


顧客にとって価値があるか、ないかで判断する。

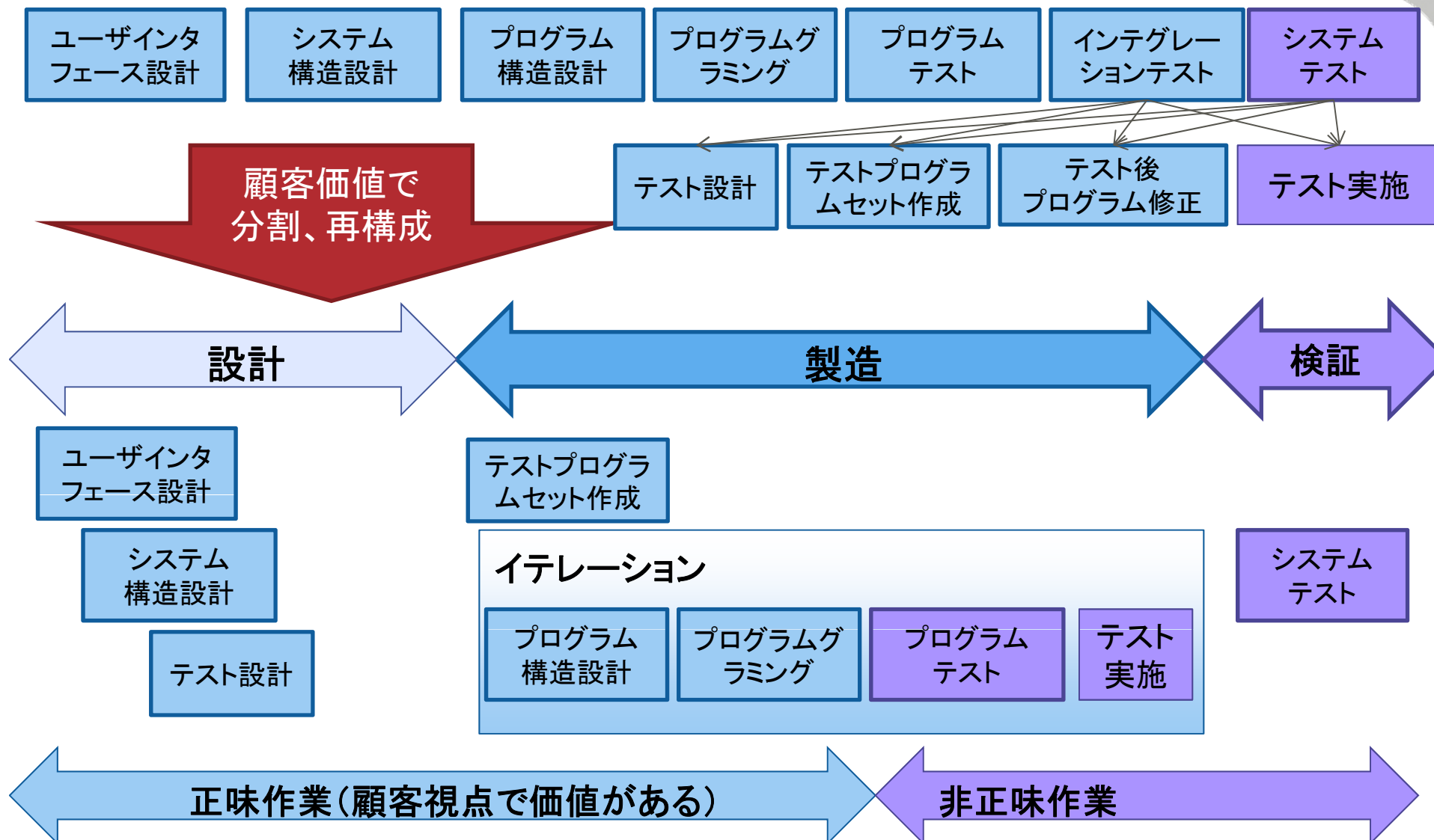
5. 開発プロセスの変化



6. プロセス改善の進み方

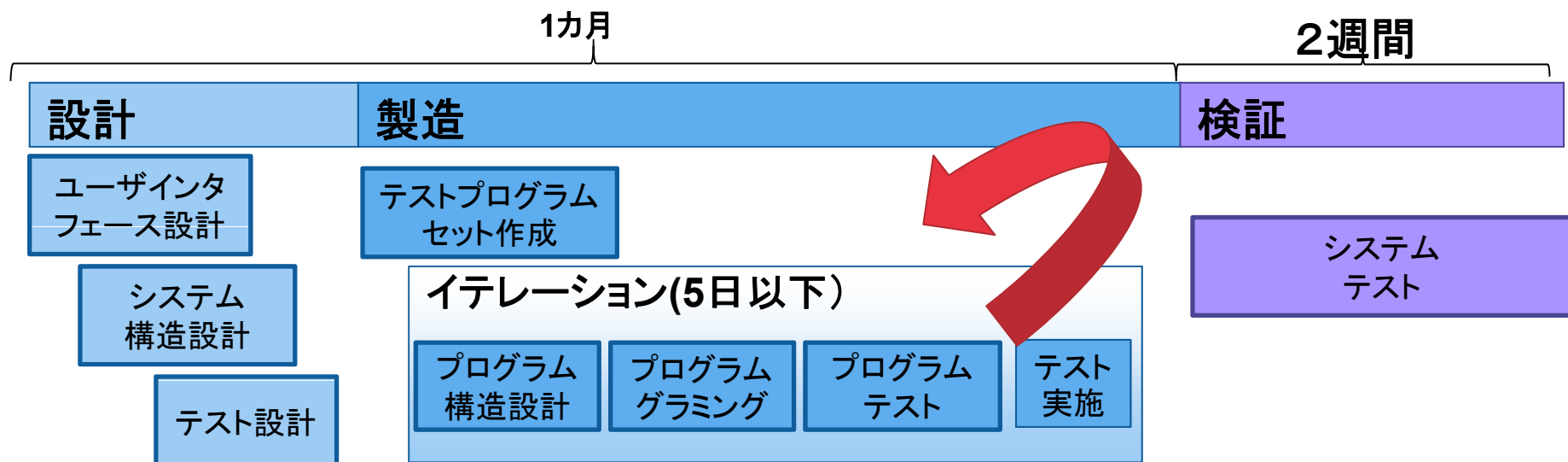


7. 顧客価値でプロセス分解



8. 制約条件(時間制約)

- 標準の製造期間を1カ月、検証期間を2週間とし、開発ラウンドとする。
- CCPMの考え方をもとにスケジュールを50%短縮し、30%をバッファとする(20%削減を目標とする)
- 製造工程は顧客価値の最小単位に分割し、5営業日以下の繰返型(イテレーション)にする。
- 開発項目ごとに目標数値との比較をし、品質評価をする。
- 開発スピード:2.0KS/人月 作込み障害件数:5件/KS



9. 開発実績



	V01L10	V01L20	V01L30	V01L40
規模	131.2KS	64.6KS	59.0KS	89.7KS
レビュー障害	11.6件	7.5件	10.1件	8.9件
テスト障害	3.3件	2.8件	2.7件	4.1件
テスト項目率	-	262件	150件	186件
開発スピード	-	-	1.2KS/人月	1.7KS/人月
開発工数			49.1人月	52.7人月
生産性(ラウンド)	-	-	16.7DIL(167DIL)	22.6DIL(272DIL)
開発外障害累積	0.42件	0.16件	0.17件	0.12件
顧客発生障害(現在)	0.01件	0.00件	0.00件	0.02件
顧客発生障害(ピーク)	0.18件	0.05件	0.05件	0.07件

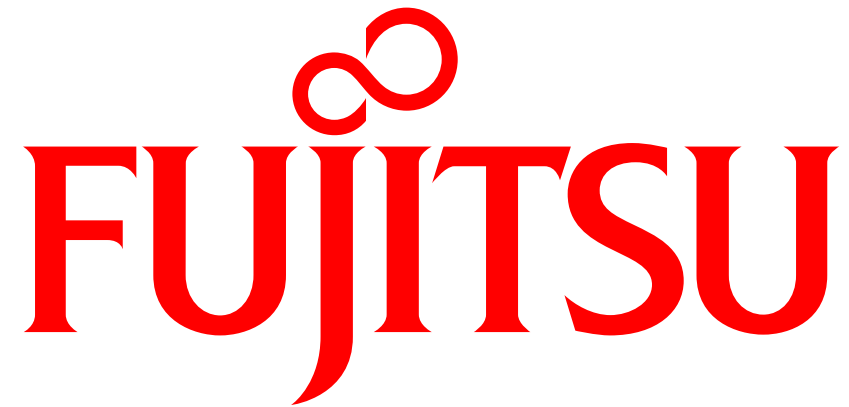
※開発期間はV01L30とV01L40とほぼ同じで、生産性が向上している。

出荷後の顧客発生障害は、**2010年1月頃をピークに減少をしている。**件数はKSあたりの数値である。

10. 開発現場からの声



- 1回の開発規模が小さいため、状況を把握しやすい。
- 開発規模を小さくすることにより、思考が集中できる。
- 要求の変化に柔軟に対応できる。無駄な機能を作らなくて済む。
- テストを簡単に構築したり、確認できるように改善していきたい。



shaping tomorrow with you

補足1 テストの考え方

- 修正内容に関係なく**全機能をテスト**——
- 繰り返しやるため、**自動テスト**
- **NGの判定**が簡単

1. テスト計画
2. テストセット作成
3. テスト実行
4. テスト結果判定
5. 障害調査

自動化

2回目以降は、
低コストで可能

レベルダウン防止
工数=0

修正に躊躇する
ことがなくなる。

だれでもできる

開発遅れなどを
テスト量で調整して
しまいがち、それだ
と品質は悪いまま。
テストの範囲を縮め
るとすぐにレベルダ
ウンする。

障害が発生して
もトレースできる。

結果が残る。
判定にばらつきが
少ない。

手動でやると状況
によってばらつき
が発生する。
テストの目的は
開発を終わらせる
ことではない。

補足2 システムテストの考え方



- 検証する目的を明確にする。

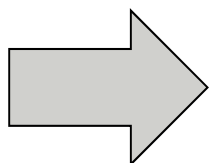
- 運用フローテスト
- 新機能確認テスト
- 障害修正全機能テスト



→ レベルダウンの防止

- テストの標準時間を規定する。(2週間)

→ 非正味であり、提供までのスピードのボトルネックになる。



設計から検証までの**最短期間は1.5カ月**
(製品提供のスピード)

補足3 障害の作込みにくい単位

FUJITSU

■ 定期的な開発状況報告

→ 期間別、会社別の品質データを集計



■ コンポーネント？ 外製と内製？ 開発言語

➤ 特性に違いはなく、集計値では差がない



■ 過去1年分のデータを明細で調査

→ 開発規模でデータをソート

50Step以下 開発スピード: 1.0-1.5KS/人月 障害0件/KS

300Step以下 開発スピード: 2.0-3.0KS/人月 障害 5件/KS

300Stepより上 開発スピード: 0.5-1.0 KS/人月 障害10-20件/KS

➤ 当てはまらないものが品質不良をおこしている。

当てはまらないものとは開発規模が300Stepより多いのに、開発スピードが早いもしくは、障害検出件数が少ないものは、ST以降（顧客先を含む）で障害を検出している。

