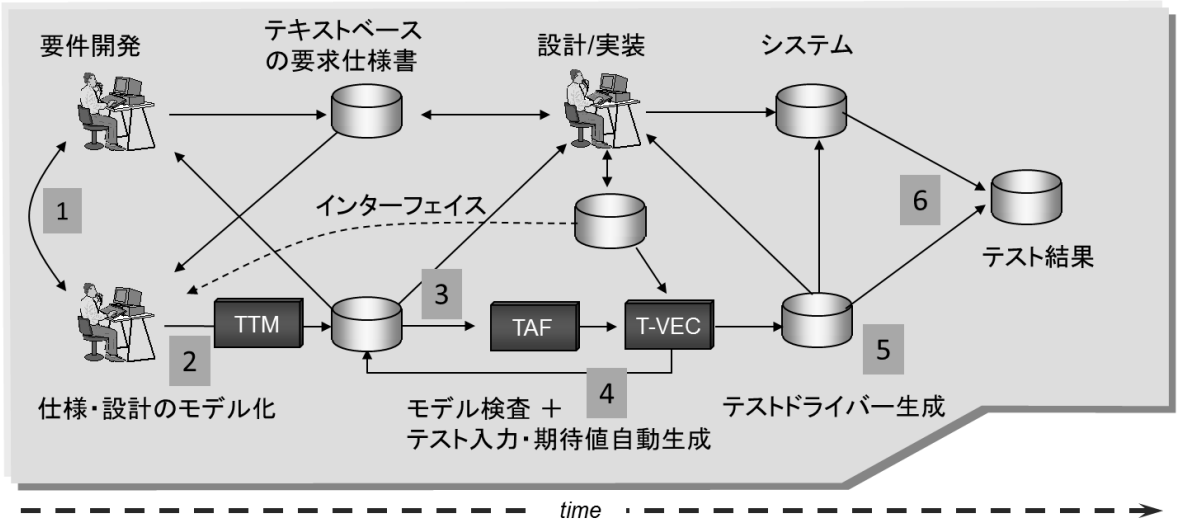
テストは人手に頼らない！

1：仕様を形式手法でモデル検査すると同時に

2：テスト入力・期待値・ドライバーを自動生成

数千行ものテストスクリプトの代わりに、早期段階から MBT を通じて要求をモデル化すれば、要求仕様を洗練させて、インターフェイスを明確に定義した疎結合な設計を育み、テストの自動化を得ることができる。派生開発ではリソースに限りがあることから、変更・追加要件のみに割り切って MBT を取り入れる。既存システムが結合状態にある場合は（殆どはこちら）、ラッパーを介したコンポーネント化などによりカップリングを排除するなど、上手な工夫も考える。

XDDP 変更三点セットで仕様変更を掌握したら MBT でテストが楽になる！

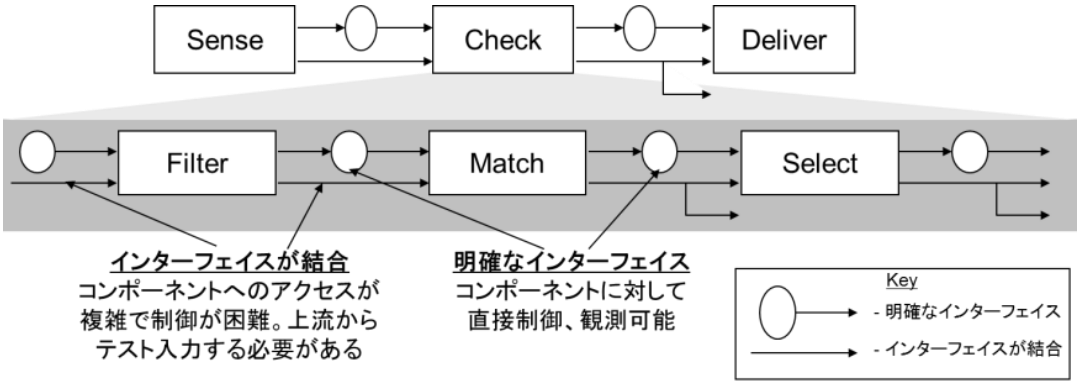


医療機器の派生開発事例では、システムレベルの機能仕様からテストシナリオをモデル化。（例えば、体内式除細動器のパラメータを変更する機能）　そしてコード設計上の詳細をモデル上の制約として加味することで、高いカバレッジを満たすテスト自動化環境を構築できた。

こんな症状でお悩みでは？

ソフトウエアに品質、安定性の問題がある
要件上の問題が多くの問題の原因となっている
テストドライバーのデバッグ工数がバカにならない
機能間の問題に対して多くのテスト工数を要している
要件トレーサビリティを求められるが、もう身が持たない

さらに詳しい情報 => http://www.fuji-setsu.co.jp/products/T-VEC/xddp_MBT.html



“既存システムの機能は、メモリスペースの制限など、過去から引き継いできた多くの制約と密接に関わっていました。また、Filter、Match、Select 間のインターフェイスは、明確に定義されていませんでした。そのため、テストプロセスは複雑でした。多くのテストは、Check のようなサブシステムに対して、上位層レベルから実行される必要が有りました。なぜなら、入力のいくつかは、Check コンポーネントから上流へセットされるからです。さらに、Match など機能からの出力は、可視的では有りませんでした。そのため、これら下位層レベルにあるコンポーネントの体系的、かつ包括的なテストは、困難でした。それゆえ、実装を介して、これら下位層レベルのコンポーネントの、各スレッドに対するテストのカバレッジを満たすためには、上位層レベルのコンポーネントからのテスト実行が増大することになります。時にはテストの数量が大規模になってしまいます。

要求仕様のモデリングを行うことで、早期段階から入出力間のインターフェイスは明らかになり、内部ステート情報を得ることも相まって、テストの容易性が飛躍的に向上しました。デザインチームにより改善されたインターフェイスへの対応により、おおよそ 80%の機能がテストされるようになりました。これにより、モデルとテストの複雑さを飛躍的に軽減し、より少ないテストで工数とコストを削減しながら、最大限のカバレッジが得られるようになりました。 ちなみに、残りの 20%は、性能や、コストへのインパクト、リソース、再テストのスケジュールなど、改善できない要素に関わるコンポーネントです” 〈T-VEC Wiki“テストを容易にするモジュール構造”より〉

T-VEC のモデルベースドテスト（MBT）は，仕様をモデル化して形式手法で検査。テストを開発と並行して進行することで、早期段階で仕様上の欠陥を排除して手戻りを削減。加えて、テスト入力・期待値・ドライバーを自動生成するので、後工程の工数を飛躍的に軽減する。

－形式手法による定理証明（人では解析できない階層間に存在する矛盾、線形・非線形）
－自動生成されるテストベクタ（入力・期待値）は上下限、境界値、線形・非線形をサポート
－フロート、ダブルなどあらゆるデータ型をサポート
－モデル化による共通部品化で再利用性向上
－モデルをベースにすることで様々なテストの成果物の構成管理を単純化
－要件等の変更にもモデルの変更のみで直ちにテストできる
－状態爆発しない（サブシステムごとに上流のコンストレインツを加味した入力で検証）

要求の仕様化は、機能安全規格や社内ルールに従うことが目的では形骸化しかねないが、
T-VEC があればテストが楽になるので嬉しい！