

ソースコード診断 & 資産化

使える図面

ソースコードから図面を作り
設計構造の改善箇所を特定

ソフトウェア資産へ

共通部のプラットフォーム化
変動点でプロダクトライン化



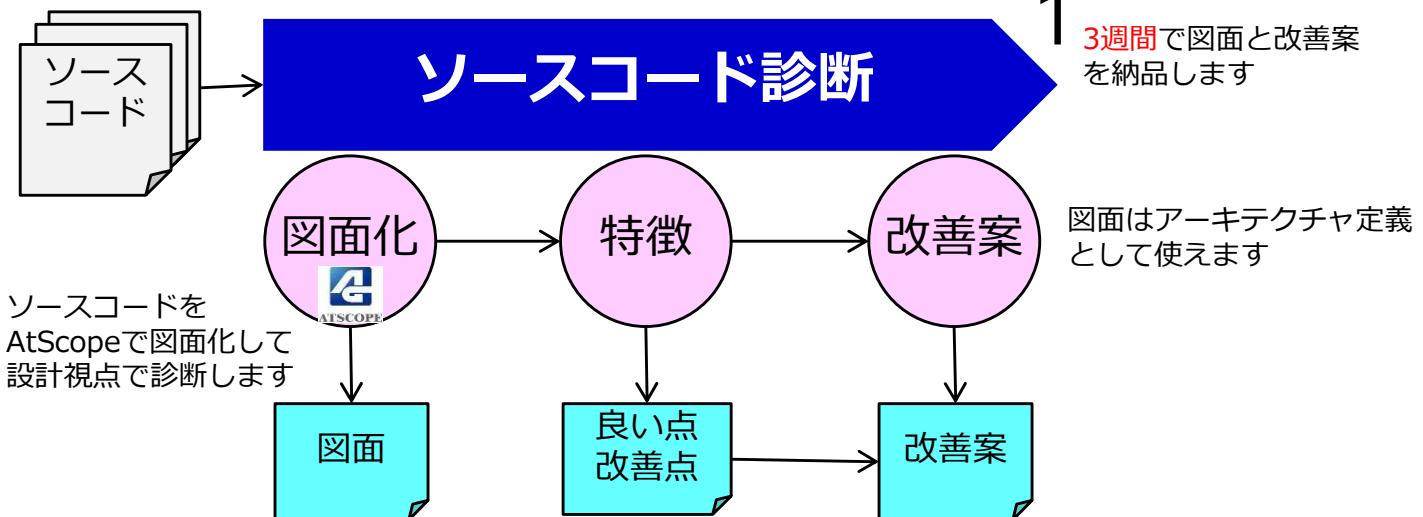
在庫を **資産** へ変換します



ソフトウェア**在庫** → **ソフトウェア資産**

ソースコードのみで構造品質の診断を行います

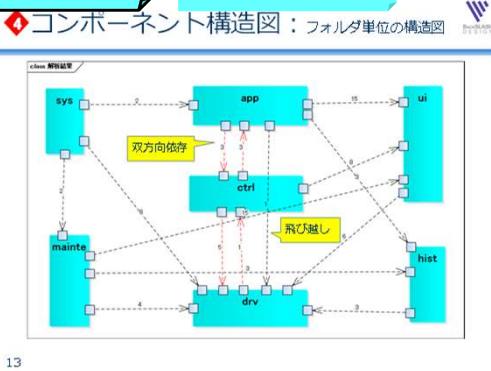
3週間



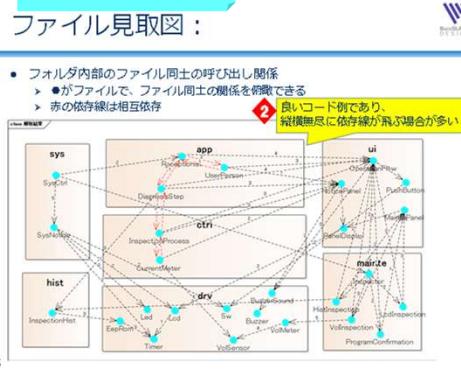
ソースコード診断の例

図面

フォルダ粒度

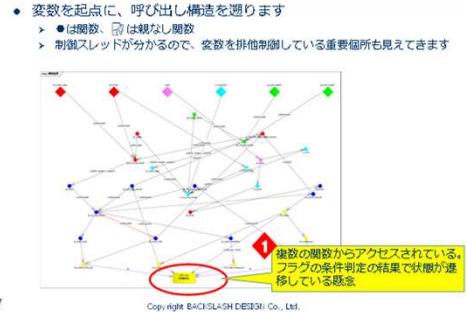


ファイル粒度



関数粒度

関数構造図（変数起点）



良い点

コードの特徴（良い点）

No	特徴	説明
1	わかりやすい名前付け	ファイル名、関数名、変数名が問題ドメインの名前になっている。とても理解しやすい。
2	短い関数	ひとつの関数が、画面スクロールしなくとも全体が見えるものが多い。單純でスムーズ。
3	部品化	部分がペーストになっている。再利用しやすいソフトウェアになっている。
4	割込み処理の分離	割込み関数がひとつのファイルに集まっている。イベント順序の違いなど、隙合の動作を追いかける。
5	規則的な処理ロジック	機能ごとにmainがあり、規則性があるため修正箇所を見つけやすい。

7

改善点

コードの特徴（改善点）

No	特徴	説明
1	大きなファイル	10,000行を超えてるファイルがある。
2	ハイブリッド結合の状態変数	状態変数の取りうる値が7を超えてる。
3	割込み内での処理	割込み処理内で長い処理をしている。
4	グローバル変数	変数がファイル外に公開されている。
5	フラットなフォルダ構成	ひとつのフォルダ内に全てのファイルがある。
6	main処理が一筆書き	mainという関数内部で、複数の処理をしている。ファンアウト数が7を超えてる。

8

改善案

改善策のご提案：トップ5

No	項目	予測される問題点	設計改善施策
①	状態変数	状態変数の取りうる値が多く、知っている人しか修正できない。変更の影響範囲の見極めが困難、かつ、コストしきれない。	状態変数を分割する。データ設計。
②	ファイル分割	内部を追いかけないと修正できない。少しおかげで影響箇所が多い。	ファイルを分割する。リファクタリング。
③	データ競合	親なし開閉箇所で共有する変数があり、イベントの発生順序によって再現性の低い障害を誘発してしまう。	初期リストードを横断する変数をガードする
④	アーキテクチャ構造	全体構造が見えてこないため、習得のリードタイムがかかる。担当者の引継ぎが困難。	フォルダ分けして、アーキテクチャ構造を明確にする。
⑤	カプセル化	グローバルデータが多いため、修正時に副作用を生じやすくなる。不具合の修正がなかなか収束しない。	クラスを構造体へ。変数をファイル内部へカプセル化する。static宣言。

10

資産化の例：コード起点プロダクトライン化

動くコード

設計意図不明

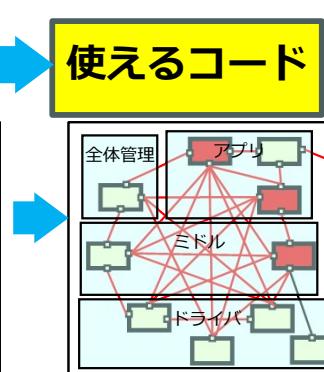


赤線：双方向依存
赤箱：複雑度高い

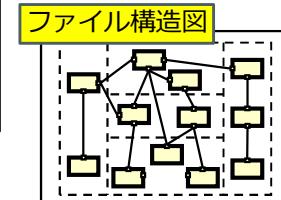
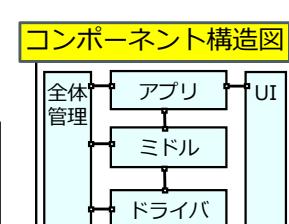
秘伝コード

使えるコード

影響範囲が分からず
品質が安定しない
想定しない不具合発生



コンポーネント単位に
括って俯瞰する

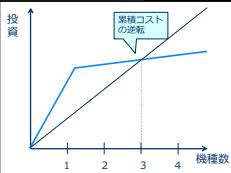


変動点定義

資産活用



投資対効果



ソフトウェア資産

- 長持ちし置換しやすいソースコード
- 全体を俯瞰するアーキテクチャ
- 変動点を特定するフィーチャ
- 変動点の機能確定時に応じた実装

サービス価格

サービス	期間	主な納品物	価格（税込）
ソースコード診断	3週間	アーキテクチャドキュメント初版 (5つのビューの図表を統合)	176万円
資産化 オフショアでの開発です	1カ月～	プラットフォーム成果物（コンポーネントAPI） プロダクトライン成果物（変動点定義）	個別見積り 99万円／月～