Web アプリケーションのユーザビリティ向上に関する提案
—EC サイトを例として—

2014SE050 北野大地　2014SE051 北山裕斗　2014SE063 三井卓弥
指導教員：野呂昌満

1　はじめに

Web ページに多様なデバイスでアクセスする機会が増えたことにより，画面表示での即応性などのユーザビリティの低下が問題となる場合がある．現在，多様なデバイスに対応した Web ページ作成の技術としてレスポンシブ web デザインがある．レスポンシブ web デザインではデバイスの画面サイズに応じて画面を構築することができるが，表示しない画像やコンテントまで読み込んでしまい，通信速度が遅いときには表示速度の向上は見込めない．

本研究の目的は，Web ページ上の不要な情報を排除し，環境に応じた必要な情報だけを画面に表示することで，Web アプリケーションのユーザビリティを向上させることである．実現の方法として，MVC の派生形である MVVM アーキテクチャの ViewModel を書き換えることと，実行効率に関わるアスペクトを定義することによって，Web ページ上の不要な情報を排除し画面表示の即応性を向上させる．コンテキスト指向では，動的にレイヤを活性化・非活性化することにより実行時にシステムの振舞いを変化させることができる．この技術を用いることで不要なコンテントを排除し，画面の表示方法を動的に変化させて実現する．

本研究室ではインタラクティブシステムのためのアーキテクチャである CSA/I-sys が提案されている．CSA/I-sys は MVC アーキテクチャに基づき，アスペクト指向アーキテクチャとして設計されている．このアーキテクチャをコンテキスト指向に基づき拡張することにより，実現の可能性を提案する．

2　背景技術

2.1　CSA/I-sys[1][3]

本研究室で提案されているアプリケーションアーキテクチャ CSA/I-sys を図 1 に示す．本研究室で提案されている CSA/I-sys はインタラクティブシステムを対象とし，様々な開発環境が混在する現状での開発支援を目的としたソフトウェアアーキテクチャパターンである．CSA/I-sys において，参照アーキテクチャは開発環境を規定し，アプリケーションアーキテクチャは実行時環境を定義するとの認識のもと，参照アーキテクチャを設計し，それを詳細化したアプリケーションアーキテクチャとして定義されている．CSA/I-sys は MVC アーキテクチャとその派生に基づき設計されている．以下に，CSA/I-sys の柔軟性とその機能を示す．

- Model
  - ビジネスロジックとデータ処理：画面構築との分離
  - 状態と状態におけるイベントに対応して実行される処理：アプリケーションの部分的な自動生成
- View
  - 内部表現と外部表現：モデルとのバインディングとその表現を分離
  - 内容，役割，見栄え：標準の Web 技術に基づく，部分的な再利用
- Controller
  - 内部イベント，外部イベント：イベントの形式を分離

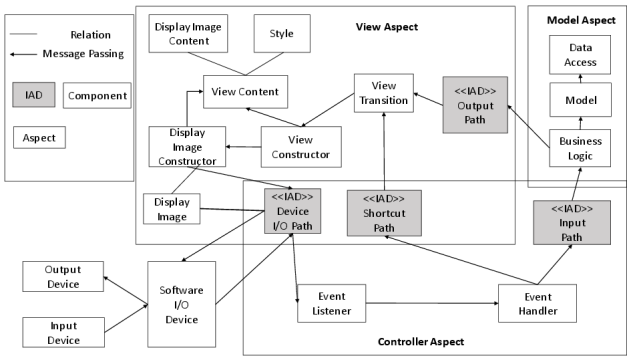

図 1　CSA/I-sys

2.2　コンテキスト指向プログラミング

コンテキスト指向プログラミング (COP: Context Oriented Programing) とは，コンテキストに依存した振舞いをモジュール化するためのプログラミング方法である [4]．コンテキストをプログラムから観測できる外部環境やシステムの状態と定義し，特定のコンテキストで実行される振る舞いの集合をレイヤーとする．コンテキストに依存する振舞いのモジュール化を実現するための要素として，主に以下の 3 つがある [5]．

- 層 (layer):特定の文脈で実行される振舞いを抽象化したもの
- 部分メソッド (partial method):特定の文脈を実現したコードの断片
- 層活性 (layer activation):層を動的に活性化，非活性化させ，現実活性な層の部分メソッドのみを実行させる機構

2.3 MVVM

MVC アーキテクチャの派生形であり, プレゼンテーションロジックとビジネスロジックの分離を目的とした MVC とは異なる分割によるアーキテクチャである．画面に表示される視覚的要素とユーザ操作によるイベントを扱う View，ドメインモデルを扱う Model, View と Model を関連づけ外部表現とドメインモデルを結合した ViewModel によって構成される．

2.4 アスペクト指向技術

アスペクト指向とは，横断的関心事（Crosscutting Concern）を単一モジュールとして分解させることで, モジュール間の独立性を高め，補うモジュール化技術である [6]．横断的関心事とは，他の関心事に横断的に関係があり，単独では取り扱えない関心事である．

3 パフォーマンスを考慮した具象アーキテクチャ

3.1 設計指針

本研究では不要な情報を Web ページ上から排除し, ユーザビリティを向上させるために，CSA/I-Sys を拡張しコンテキストに応じて動的に Web ページの再構成を行う方法を考案する．

通信速度に応じて適切なコンテントだけを読み込み表示する処理と，画像の解像度を下げて表示する処理を行うアーキテクチャを設計することでユーザビリティの向上を図る．ここでコンテントの表示と非表示を切り替える処理を行うために Model 自身を変更してしまうと, アプリケーション自体を書き換えてしまうこととなり，構造が複雑になったり実行時間が長くなってしまう．この問題を解決するために，我々は CSA/I-sys に MVVM の概念を導入し，Model 自身は変更せずに ViewModel を動的に再構成をすることで解決する．さらに，状況に応じて ViewModel を動的に再構成して最適な画面構築を行うためにコンテキスト指向の概念を導入することで柔軟にポリシー変更可能な構造を目指した．

コンテキストと実行効率を考慮した場合，コンテキストに応じた振舞いの変更処理や，コンテントの表示と非表示の切り替え処理が各所に横断する．我々はアスペクト指向技術を適用し、コンテキストと実行効率をアスペクトとして分離する．分離したアスペクトをそれぞれ ContextAspect，PerformanceAspect と定義し、これらを統一的に扱うアーキテクチャを提案する．

3.2 ContextAspect の定義

コンテキスト指向を考慮してコンテキストに応じた振舞いの変更を可能にするために，ContextAspect を定義した．ContextAspect は,

- Context
- LayerActivator
- Layer

の 3 つのコンポーネントから構成される．Context が EventListener から必要なコンテキスト情報をフックし，そのフックされた情報によって LayerActivator が Layer の活性と非活性を切り替えることでコンテキストに応じた動的な処理の変更を可能にする．Layer にはコンテキストごとの振舞いが記述されている．

3.3 PerformanceAspect の定義

ViewModel を書き換えて View の再構成を可能にする PerformanceAspect を定義した．PerformanceAspect 内で定義される PerformanceManager は，Layer の記述に応じて，ViewConstructor の構造を変更する．それにより ViewModel を書き換えて Web ページの動的な再構成が可能となる．

3.4 パフォーマンスを考慮した具象アーキテクチャの再設計

使用されるデバイスや通信速度などの環境の違いによって表示するコンテントの切り替えや画像の変換を行うために，3.1 と 3.2 で定義した ContextAspect と PerformanceAspect を CSA/I-sys に導入した．また，CSA/I-sys では ViewAspect 内にあったコンポーネント DisplayImageContent，Style，ViewContent，ViewConstructor，ViewTransition を ViewModelAspect として分離した．アスペクトを導入した CSA/I-sys を図 2 に示す．

これによって ViewModelAspect を View 側に配置するか Model 側に配置するかでコンテント削減と画像変換の処理が可能となる．

ViewModelAspect を View 側に配置した場合，PerformanceAspect がコンテキストに応じて必要な画像だけを PULL 型で Model から読み込み，ViewConstructor の書き換えを行う．PUSH 型と PULL 型の両方を IAD(アスペクト間記述) に記述しておき, その両方の API(Application Programming Interface) を Model と ViewModel が持っていることを前提とする．PUSH 型と PULL 型を PerformanceAspect が切り替えることによって，Model 自身の変更を行うことなくコンテントの表示を変更し動的に画面を再構成することが可能になる．ViewModelAspect を Model 側に配置した場合，PerformanceAspect がコンテキストに応じて画像解像度の変換命令を送る．それによって ViewModelAspect で画像を変換し ViewAspect へ送る．これによって通信速度が遅いときにデータ容量の大きい画像の読み込み時間を短縮でき，ユーザビリティの低下を防ぐことが出来る．

4 コンテキストとポリシーの定義

本研究では，デバイスの通信速度が低下した場合と，スマートデバイスなどの小さな画面で EC サイトを閲覧する場合を例として検証する．通信速度をコンテキストとして定義し，考察する．Web ページ閲覧において通常考え

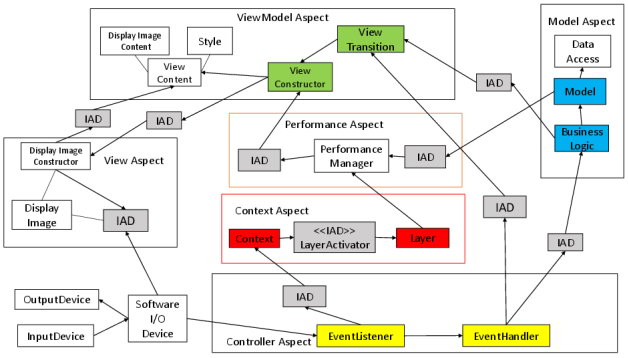

図 2　パフォーマンスを考慮した CSA/I-sys

られる快適・普通・不快の 3 段階でポリシーを定義する．一般に Web ページを快適に閲覧することが出来る通信速度 1Mbps を快適，読み込みに数秒の時間を要するとされる通信速度 0.4Mbps を普通，スマートデバイスなどで通信制限がかかり著しく通信速度が低下した場合の通信速度 128Kbps を不快とし，この 3 つを基準として Layer を切り替える．

通信速度	画像	コンテンツ
1Mbps	画像全表示	全表示
0.4Mbps	画像一部解像度低下	一部表示
128Kbps	画像解像度低下	一部表示

図 3　定義したポリシー

分ける段階はいくつであっても，コンテキストの書き換えによってソフトウェア自身を作り替えることができる．段階にあまり意味はなく，ここでは最低限の分け方をした．動作例を図 4 に示す．

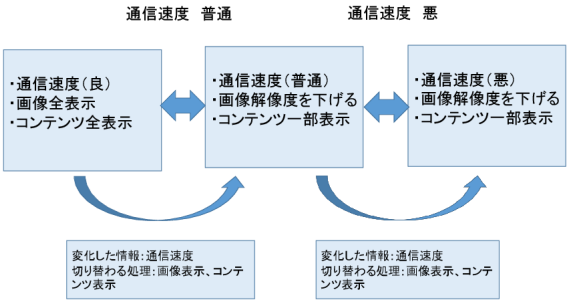

図 4　コンテキストに応じた動作例

5　事例による考察

5.1　コンテキストに応じた振舞いの変化の例

コンテキストに応じたレイヤの振舞いの変化の例を図 5 に示す．EventListener が通信速度を計測し，コンテキストにイベント通知を行う．Context がコンテキストの状態を推定し，LayerActivator に更新通知をする．LayerActivator がコンテキストの状態に応じた Layer の活性化を行う．図の例では通信速度が快適なら HighSpeedLayer を，通信速度が不快なら LowSpeedLayer を活性化させる．そして活性化された Layer がコンテキストの状態に応じた振舞いを PerformanceManager にイベント通知を行う．図の例では High Speed Layer が活性化した状態ならばコンテントを全て表示し，Low Speed Layer が活性化した状態ならば一部のコンテントを表示しないようにする．

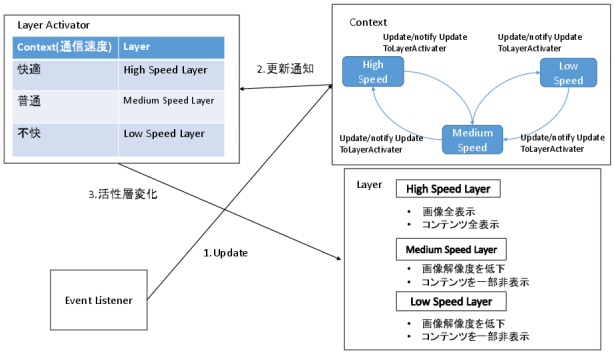

図 5　コンテキストに応じたレイヤの振舞いの例

5.2　画像変換

通信速度が遅いときに容量の大きな画像の読み込みを行うと表示までに時間がかかり画面表示のユーザビリティが低下してしまう可能性がある．その場合，画像の解像度やサイズを適切なものに変換して画面の再構築を行う．図 6 に画像変換を行う場合のアーキテクチャの振舞いを示す．

User からのデバイス情報が EventListener へ送られたとき、コンテキストに関わる情報をフックし，コンテキストに応じて ContextAspect が PerformanceAspect の振る舞いを変える．そこで画像を変換する必要がある場合には，ViewModelAspect を Model Aspect 側に配置し，ViewConstructor が ViewModel の構築ロジックを変更する．DisplayImageConstructor が Model と ViewModel の情報を元に画像を変換し，DisplayImage を生成する．

5.3　コンテント削減

画面サイズに対して，容量が過多な場合ユーザビリティが低下してしまう可能性がある．その場合，表示の優先度が低いコンテントを非表示にして画面の再構築を行う．図 7 にコンテントの表示切り替えを行う場合のアーキテクチャの振舞いを示す．

User からのデバイス情報が EventListener へ送られた

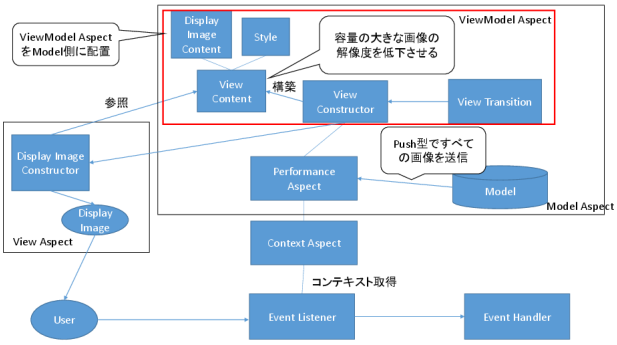

図 6　画像変換を行う場合のアーキテクチャの振舞い

とき、コンテキストに関わる情報をフックし，コンテキストに応じて ContextAspect が PerformanceAspect の振舞いを変えるまでは画像変換の場合と同じ振舞いをする．そこで不要なコンテントを非表示にする必要がある場合には，Model から受け取ったコンテントの中から Layer に記述された表示優先度の高いコンテントのみを PerformanceAspect が判断し，Model からのデータ取得を PUSH 型から PULL 型に切り替え，必要なコンテントの取得し，ViewConstructor へ送信する．そして画像変換の時と同様に DisplayImageConstructor が DisplayImage を生成する．

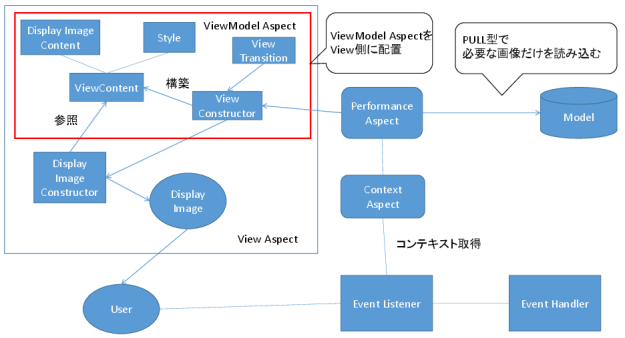

図 7　コンテントの表示切り替えを行う場合のアーキテクチャの振舞い

5.4　レスポンシブ web デザインとの比較

レスポンシブ web デザインでは 1 つの HTML を CSS で制御し，ユーザが閲覧するデバイスの画面サイズに応じてページのレイアウト・デザインを最適化して表示させる技術である [2]．コンテントをすべて読み込む必要があるので，通信速度が遅いときに容量の大きな画像を読み込まなければならず即応性が低下してしまう．

本研究では，PerformanceAspect を導入することによって通信環境が悪くなったときにモデルからのデータ取得を PUSH 型から PULL 型に変更し，必要なデータのみをモデルから取得する．これによりレスポンシブ web デザインではできなかったモデルを変更しないで使用者に不必要なコンテントを減らすことができ即応性を向上することが出来る．

6　おわりに

本研究では，web ページの画面表示のユーザビリティに着目し，多様なデバイスに合わせて適切な画面構築を動的に行うために，コンテキスト指向に基づき CSA/I-sys の拡張を行った．今回は通信速度が低下した場合でも快適な Web ページの閲覧を行うことができるように CSA/I-sys に ContextAspect と PerformanceAspect を導入し，ViewModelAspect を分離した．ViewModelAspect を View 側に配置することにより，Model を書き換えることなくコンテントの表示を切り替えて View を動的に再構成することを可能にした．また，ViewModel Aspect を Model 側に配置することにより，Model を変更することなく画像の解像度を下げて View を動的に再構成することを可能にした．

また，コンテキスト指向とアスペクト指向を考慮することによって，コンテキストを書き換えるだけで柔軟にポリシーを変更することが可能になった．

今後の課題として，画面表示を行わない不要な情報を判断するための仕組みを実装することが必要である．考えられる方法としては使用者のコンテントへのアクセス数やページの表示時間などの情報をデータベースから取得し，コンテントに優先度をつけるなどが挙げられる．

参考文献

[1] A. ESAKA, M. NORO, and A. SAWADA, “Design of Common Software Architecture as Base for Application Generator and Meta-Generator for Interactive Systems,” *IEEE*,vol.2, pp.323-328, 2017.

[2] Bohyun Kim, “Responsive Web Design, Discoverability, and Mobile Challenge, ” *Library Technology Reports*, vol.49, no.6, pp.29-39, 2013.

[3] 江坂篤侍, 野呂昌満, 沢田篤史, “インタラクティブシステムのための共通アーキテクチャの設計, ” ソフトウェア工学の基礎ワークショップ論文集，vol.24，no.43，pp.129-134，2017.

[4] 紙名哲生，“文脈指向プログラミングの要素技術と展望，” コンピュータソフトウェア，vol.31，no.1，pp.3-13，2014.

[5] 紙名哲生，青谷知幸，増原英彦，玉井哲雄，“ユースケースを用いた文脈指向ソフトウェア開発，” ソフトウェアエンジニアリングシンポジウム，pp.1-8，Sep. 2011.

[6] 千葉滋，アスペクト指向入門-Java・オブジェクト指向から AspectJ プログラミングへ，技術評論社，2005.