コンテキストを考慮したインタラクティブソフトウェアのための共通アーキテクチャ

2013SE009 青山黎明
指導教員：沢田篤史

1　はじめに

近年，複数のセンサの搭載された組込みシステムの高機能，多機能化が進んだ．このような組込みシステムはセンサ情報により，周囲の環境，外部の状況の変化を認識できる．アクタやシステム、環境などの変化をコンテキストとし，コンテキスト情報に応じ振舞いを変化させるコンテキストアウェアなシステムが実現可能となった．このようなシステムに対し，コンテキストに依存する振舞いをモジュール化するコンテキスト指向技術が注目されている．一方，スマートデバイスなどのインタラクティブソフトウェアも同様，高機能・高性能化にともない，設計効率化や保守性向上が求められている．

インタラクティブソフトウェアにおいてもコンテキストを考慮する必要がある．例えば，多様化するデバイスに応じて適切な画面表示を行う振舞いや，室内や室外などの状況に応じて取得する地図を変更する状況に基づいてモデルを切り替える振舞いなどである．このような状況に依存する振舞いをオブジェクト指向で設計すると，振舞いを規定するプログラムが各コンポーネントに散在することで保守性，変更容易性の低下につながる．また，様々な開発環境や実行時環境があるインタラクティブソフトウェアにおいてコンテキストを体系的に扱うことは困難である．

本研究の目的は，インタラクティブソフトウェアにおけるコンテキストを考慮した開発の技術支援である．インタラクティブソフトウェアにおけるコンテキストを考慮したアーキテクチャを設計することで目的の達成を目指す．

本研究室で提案されているインタラクティブソフトウェアのための共通アーキテクチャを基本設計に，コンテキストを考慮したインタラクティブソフトウェアのためのアーキテクチャを設計する．インタラクティブソフトウェアにおいてコンテキストに依存する振舞いを分類し，それぞれ事例ごとに規定されるアスペクトをコンテキスト指向を適用することにより設計し有用性を明らかにする．

2　背景技術

2.1　コンテキス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 [2]．コンテキストとは，プログラムから観測することのできる外部環境やシステムの内部状態で，時間とともに変化し，それがプログラムの様々な実態の実行に影響を与えるものを指す．

2.2　アスペクト指向

アスペクト指向とは，横断的関心事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 [3]．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．

2.3　インタラクティブソフトウェアのための共通アーキテクチャ

インタラクティブソフトウェアのための共通アーキテクチャとは，任意の開発環境に用いて任意の実行時環境上で稼動するアプリケーションの作成支援の枠組みの基礎を与えることを目的としている [1]．共通アーキテクチャは共通参照アーキテクチャと共通アプリケーションアーキテクチャとして定義している．共通アーキテクチャは MVC アーキテクチャとその派生に基づいて設計されており，データを管理する Model，見栄えを定義する View，処理を制御する Controller に分離されている．共通アプリケーションアーキテクチャは共通参照アーキテクチャを詳細化したものであり，共通アプリケーションアーキテクチャを図 1 に示す．

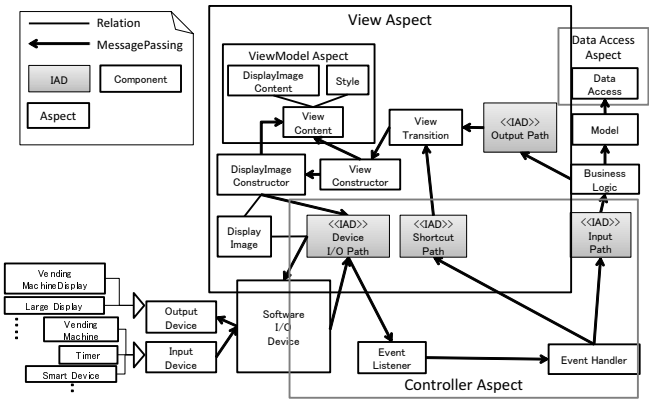

図 1　共通アプリケーションアーキテクチャ

3　コンテキストを考慮したアーキテクチャ

3.1　コンテキストの定義

コンテキストを物理量から定義される概念とし，コンテキストに応じて変化する振舞いをモジュール化したものをレイヤとする．

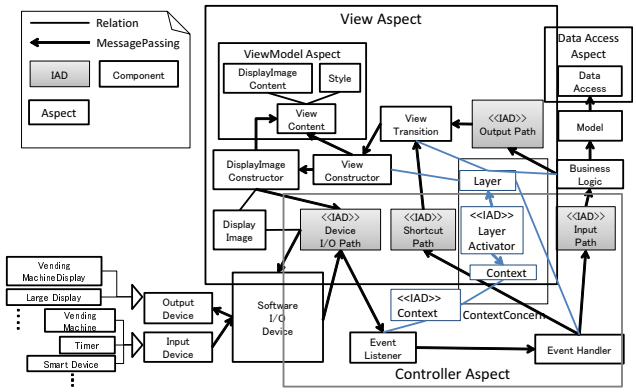

図 2　コンテキストを考慮した共通アプリケーションアーキテクチャ

3.2　コンテキストを考慮したアーキテクチャ設計

図 2 はコンテキストを考慮したインタラクティブソフトウェアのためのアーキテクチャである．多様な実行時環境や開発時環境を説明している共通アプリケーションアーキテクチャを基本設計に，コンテキストコンサーンを導入した．コンテキストコンサーンは，Context，LayerActivator，Layer の 3 点のコンポーネントで構成される．EventListener に送られてくるコンテキスト情報をフックすることで実現する．LayerActivator がコンテキストとレイヤの対応関係を管理している．レイヤが関わるコンポーネントは，

- BusinessLogic：Model のロジックを変更する
- ViewTransition：画面遷移を変える
- ViewConstructor：画面の構築を行う
- EventHandler：受理するイベントを変える

である．コンテキストによる振舞いは上記の 4 点のコンポーネントによる振舞い，またはその組合せで実現できる．

4　事例に基づく考察

4.1　View と Controller に関わる事例

画面サイズに応じて適切な画面表示を行うアプリケーションを題材とする．スマートデバイスはデバイスサイズが多様化し，各デバイスごとに適切な画面の表示形式が異なる．本アプリケーションは，デバイスの画面のスペックに応じて適切な画面表示及び画面遷移の最適化を行う．

コンテキストに対応する振舞いに着目して提案したアーキテクチャに当てはめると以下のように分類し設計できる．画面サイズを大画面，中画面，小画面に分類しそれぞれ最適な画面表示を切り替える．各コンポーネントの関係を図 3 に示す．

- コンテキスト：画面サイズ情報
- レイヤ：SmallView，MediumView，LargeView
  レイヤの振舞い
  画面サイズごとの適切な画面表示，画面遷移，受付イベントの変更
- レイヤが関わるコンポーネント
  ViewTransition，ViewConstructor，EventHandler

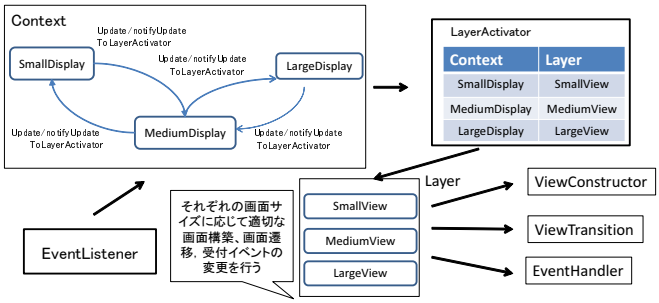

図 3　適切な画面表示に関するコンテキストコンサーン

4.2　Model に関わる事例

地図表示アプリケーションを題材とする．このアプリケーションは利用者が室内にいるときはフロアプランを表示し，室外にいるときは街路図を表示する機能を持つ．

提案したアーキテクチャを用いてコンテキストごとの振舞いを抽出した結果、以下のように体系的に表現できる．

- コンテキスト：室外，室内
- レイヤ
  室外レイヤ：地図として街路図を表示
  室内レイヤ：フロアプランを表示
- レイヤが関わるコンポーネント：BusinessLogic

4.3　考察

提案するアーキテクチャの検証として View と Controller に関わる事例と Model に関わる事例の設計を行った．結果，コンテキストに対応する振舞いを体系的に扱うことができ可読性，変更容易性の向上が期待できる．

5　おわりに

本研究では，インタラクティブソフトウェアにおいてコンテキストが依存する振舞いが関わるコンポーネントを分類し，コンテキストを考慮した共通アーキテクチャとして定義した．本アーキテクチャに基づいて地図表示アプリケーションと適切な画面表示を行うアプリケーションを設計した．今後の課題は，様々な事例によるアーキテクチャの洗練と多様な実行環境による実装を行うことである．

参考文献

[1] 江坂篤侍, 野呂昌満, 沢田篤史,“ インタラクティブソフトウェアの共通アーキテクチャの提案, ” 情報処理学会研究報告，ソフトウェア工学報告，vol. 2015-SE-187, pp. 1-8，2015.

[2] 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. 1，2014.

[3] 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.