複数の入力機器や複数の出力機器が存在する会話型システムのコンテキスト指向アーキテクチャの設計

2013SE006 青井雅英　2013SE025 萩原隆広　2013SE202 高田陽太
指導教員：野呂昌満

1　はじめに

近年，スマートデバイス上のアプリケーションと様々なセンサやアクチュエータ等がインターネットを介して連携されるようになってきた．このような状況では，表示内容を複数のディスプレイに表示したり，複数の装置からの入力を受け付けて動作することが必要になる．この典型的な例として，スマートデバイスと車載テレマティクス装置の連携がある．

本研究室では，スマートデバイスアプリケーションなどの会話型システムのための共通アーキテクチャ (以下，共通アーキテクチャ) を提案している [8]．会話型システムに複数の入出力装置を接続した環境においては，以下の機能を追加する必要がある．

- 状況に応じた出力先の切り替え
- 複数のイベント通知等を受けた際のイベントの適切な取り扱い

本研究の目的は，複数の入出力装置が追加されたり，削除されたりする場合の会話型システムのコンテキスト指向ソフトウェアアーキテクチャを提案することである．このアーキテクチャに基づいて，入出力装置の接続状況に応じた適切な振る舞いを実現することを目指す．

出力先の切り替えや入力イベントの同期は横断的関心事となる．これらをアスペクトとして分離し，共通アーキテクチャを洗練する．このアスペクトは，動的なウィーブを可能にすることで，変化する機器間の接続状況へ対応可能にする．

車載テレマティクス装置を事例とし，共通アーキテクチャと本研究で提案するアーキテクチャに基づく実現を比較し，以下を確認した．

1. 変更に対するコードの柔軟性
2. 複数の入出力装置との協調に関連する記述は，複数のコンポーネントに局所化できること
3. それらを動的に書き替えることで環境に応じた適切な振る舞いの実現可能性

2　背景技術

2.1　共通アーキテクチャ

会話型システムを前提としたソフトウェア開発の生産性の向上を目的として開発したアーキテクチャを，本研究室では共通アーキテクチャと呼ぶ (図 1)．会話型システムを前提とした抽象度の高いアーキテクチャであり，MVC アーキテクチャ (Model，View，Controller) に基づいている．

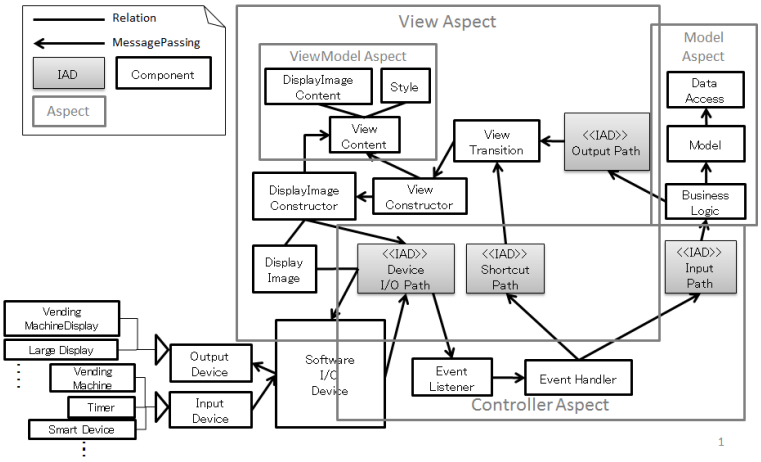

図 1　会話型システムの共通アーキテクチャ

以下に，共通アーキテクチャが担保する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
- Model，View，Controller 間の連携
  - プッシュ型，プル型の切換え：既存の実装技術はどちらかに固定

2.2　多段フィードバックキュー

短いジョブの優先，対話型プロセスの優先，また，プロセスの性質を迅速に把握し，それに基づいてスケジュールを行う．これらの設計目を満たすように意図される．複数の FIFO キューで構成されており，先頭の FIFO が最も優

先される．ディスパッチャは上位の FIFO キューから実行可能プロセスを探していき，最初に見つかったプロセスを実行する．多段フィードバックキューでは，プロセスはあるレベルのキュー上で実行される機会は，クォンタムを使い切る場合 1 回しかなく，即座に 1 つ下のレベルのキューに行くことになる．対話型プロセスはクォンタムを使い切ることが滅多にないので高いレベルのキューに存在し続けることができる．

2.3 優先度付キュー

優先度付キューは以下の 4 つの操作をサポートする抽象データ型である．

- キューに対して要素を優先度つきで追加する
- 最も高い優先度を持つ要素をキューから取り除き，それを返す
- 最も高い優先度を持つ要素を取り除くことなく参照する
- 指定した要素を取り除くことなく優先度を変更する

連想配列を用いて，それぞれの優先度に要素のリストをつなげることで実装する．

3 複数の入力機器と複数の出力機器が存在するアーキテクチャの設計

複数の入力機器と複数の出力機器が存在する場合，アプリケーションでは以下を実現する必要がある (図 2)．

- 出力先の切り替え
- 複数のイベント通知等を受けた際，イベントの適切な取り扱い

複数の機器に用意されるコンポーネントを独自に定義し，それらを組み合わせるアスペクトモジュールを別のコンポーネントで記述する．適切な組み合わせを動的に選択することで実現する．

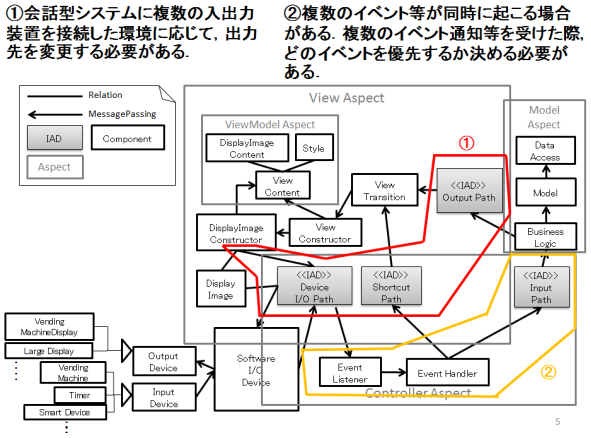

図 2 複数の入出力機器が存在する場合の共通アーキテクチャの課題

3.1 スイッチャーとシンクロナイゼーションアスペクトの織り込み

出力先の切り替えや入力イベントの同期に関連する記述は，横断的関心事になることから，アスペクト指向技術を適用することでスイッチャー，シンクロナイゼーションアスペクトとして分離する (図 3)．スイッチャー，シンクロナイゼーションアスペクトは接続状況に応じて振る舞いが変化することから，コンテキスト指向技術を適用することでコンテキストとレイヤとしてモジュール化する．シンクロナイゼーションアスペクトは，Device I/O Path から複数のイベント通知を受け取った際，どちらのイベントを優先するかを適切に取り扱う役割を持つ．スイッチャーは，Output Path，または Device I/O Path から受け取ったメッセージを，適切なディスプレイに表示する役割を持つ．

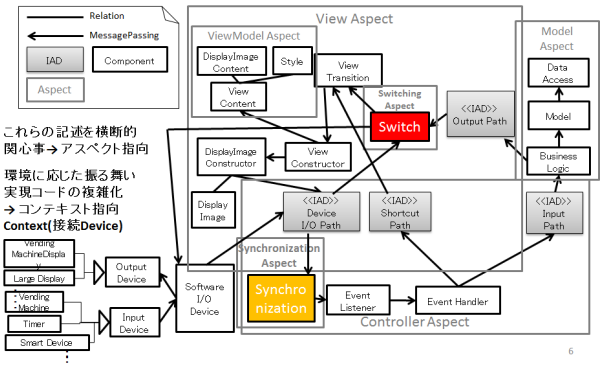

図 3 スイッチャーとシンクロナイゼーションの織り込み

3.2 シンクロナイゼーションアスペクトの設計

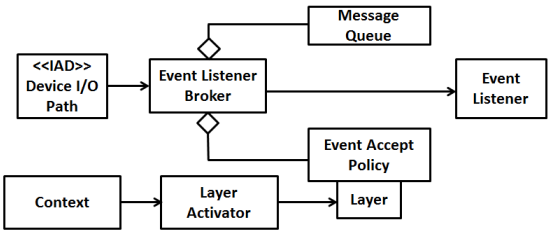

図 4 シンクロナイゼーション

Event Listener Broker に送られてきたデータを Message Queue に保管する (図 4)．受け取ったイベント通知

が 1 つの場合，このイベントは Event Listener に送られる．受け取ったイベント通知が複数の場合，Event Accept Policy が適切な取り扱いを判断する．非同期通信を行うにあたり，Message Queue を導入する必要があると考え，この構造を採用した．シンクロナイゼーションのクラス図を図 5 に示す．

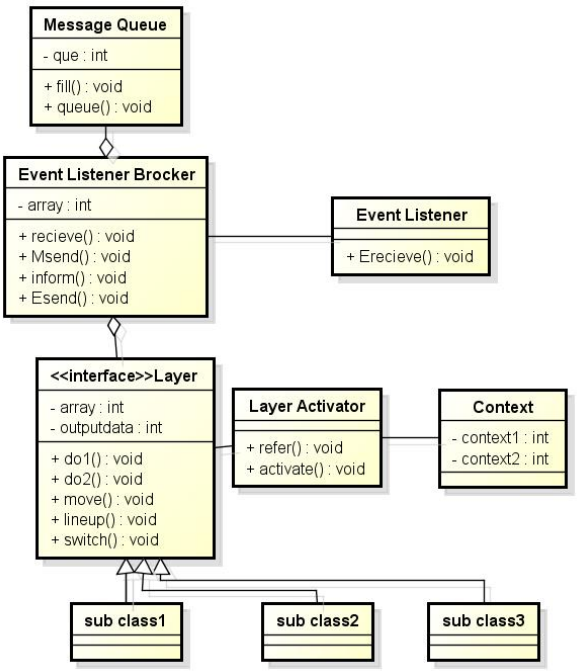

図 5　シンクロナイゼーションのクラス図

3.3　スイッチャーの設計

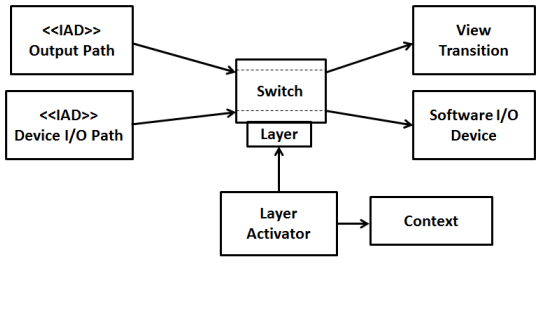

図 6　スイッチャー

Layer Activator が Context を参照して送信先を変化させる (図 6)．1)Output Path からのメッセージ，2)Device I/O Path からのメッセージ，3) 双方からのメッセージ，それぞれを 1)View Transition，2)Softwere I/O Device，3) 双方同時といったように case 文によって判断し，動的にウィーブすることで適切な振る舞いをする．スイッチャーのクラス図を図 7 に示す．

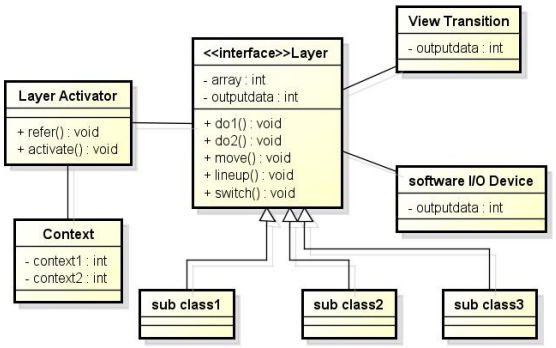

図 7　スイッチャーのクラス図

4　事例検証

4.1　複数端末への接続状況に応じた振る舞い定義

車載テレマティクスアプリケーションを題材として複数の入出力が存在する事例について考察する．車載テレマティクス装置とスマートデバイスの連携は，コンテキストに応じて出力先が変化する．物理コンテキストを距離，概念コンテキストを車内，車外とした場合の仕様と状態遷移機械を図 8 に示す．

仕様

- 出力する表示ディスプレイが変化する
- 距離（車内/車外）によって出力する表示ディスプレイが次のように変化
  - 車内：車載テレマティクス装置のディスプレイに表示
  - 車外：スマートデバイスのディスプレイに表示

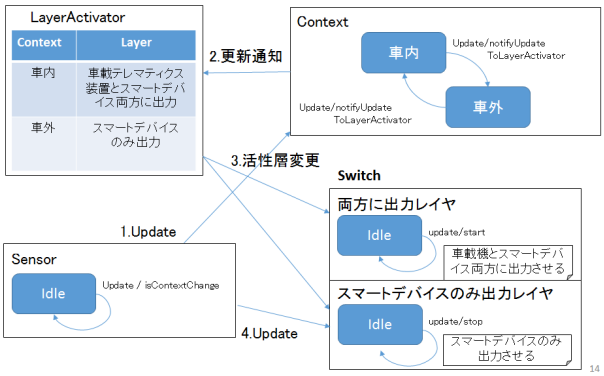

図 8　車載テレマティクス装置のコンテキストに応じた振る舞い

図 8 では Sensor が距離 (車内/車外) を判断する．コンテキストである距離 (車内/車外) が LayerActivator へ更

新される．LayerActivator は Context に応じてメッセージの送り先である Layer を変化させる．車内であれば，車載テレマティクス装置，車外であればスマートデバイスにメッセージを送る．

5　設計したアーキテクチャの有用性に関する考察

シンクロナイゼーションアスペクトを導入したことで，Message Queue を導入し，非同期通信の実現が可能になる．スイッチャーを導入したことで，状況に応じた出力先の変更が可能になる．また，アスペクトとして分離し，コンポーネント毎のプログラムの修正が可能になるので入出力機器が増えた時のプログラムの変更箇所の特定が容易になる (図 9)．

このアーキテクチャに基づいて実現することにより，変更の柔軟性が高まり，保守性の向上に繋がると考えた．

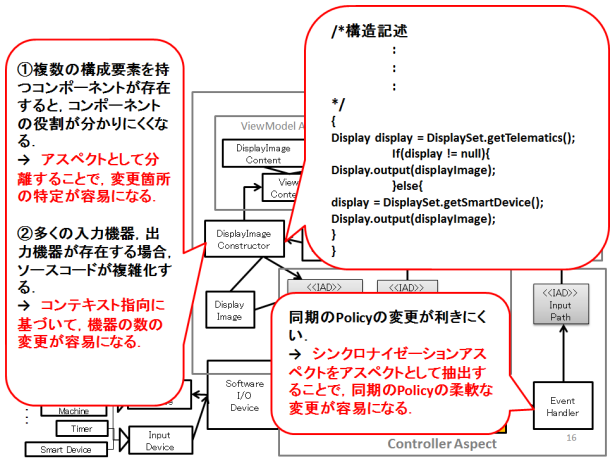

図 9　設計したアーキテクチャの有用性に関する考察

6　まとめ

会話型システムの多様化や車載テレマティクス装置の高性能化により，スマートデバイス上のアプリケーションと様々なセンサやアクチュエータ等がインターネットを介して連携されるようになってきた．表示内容を複数のディスプレイに表示したり，複数の装置からの入力を受け付けて動作することが必要となっている．具体的には，状況に応じた出力先の切り替えや，複数のイベント通知の際のイベントの優先順位の判断等の機能を横断的関心事として追加する必要がある．

本研究では，これらをアスペクトとして分離し，動的なウィーブを可能にすることでスマートデバイスなどの会話型システムのための共通アーキテクチャを洗練した．

また，車載テレマティクス装置を事例とし，共通アーキテクチャと本研究で提案するアーキテクチャに基づく実現を比較し，変更に対するコードの柔軟性を確認できた．複数の入出力装置との協調に関連する記述は，複数のコンポーネントに局所化された．それらを動的に書き替えることで環境に応じた適切な振る舞いが実現可能であることが確認できた．

今後の課題は，我々の提案したアーキテクチャに基づきコードレベルでの実装を行うことである．

参考文献

[1] M. Appeltauer, M. Haupt, and R. Hirschfeld, “ContextJ: Context-oriented programming with Java,” *Information and Media Technologies*, vol. 6, no. 2, pp. 399-419, 2011.

[2] L. Bass, P. Clements, and R. Kazman, *Software architecture in practice*, Addison Wesley, 2007.

[3] R. Hirschfeld, P. Costanza, and O. Nierstrasz, “Context-oriented programming,” *Journal of Object Technology*, vol. 7, no. 3, pp. 125-151, 2008.

[4] G. Salvaneschi, C. Ghezzi, and M. Pradella, “Context-oriented programming: A software engineering perspective,” *Journal of Systems and Software*, vol. 85, no. 8, pp. 1801-1817, 2012. 1801-1817.

[5] K. Sokolova, M. Lemercier, and L. Garcia, “Towards High Quality Mobile Applications. Android Passive MVC Architecture,” *International Journal On Advances in Software*, vol. 7, no. 2, pp. 123-138, 2014.

[6] H. V. Vliet, *Software engineering: principles and practice*, Wiley, 2007.

[7] 江坂篤侍, 野呂昌満, 沢田篤史, 繁田雅信, 谷口弘一,“ コンテキストアウェアネスを考慮した組込みシステムのためのアスペクト指向アーキテクチャの適用と実現, ” ソフトウェア工学の基礎ワークショップ（FOSE2016）論文集，vol.23，pp.175-180，2016.

[8] 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, no.32，pp.1-8，2013.

[9] 紙名哲生，青谷知幸，増原英彦.“ 文脈指向言語 EventCJ への合成層の導入, ” 情報処理学会プログラミング研究会，2012.

[10] 千葉滋.“ アスペクト指向ソフトウェア開発とそのツール, ” 情報処理，vol.45，no.1，pp 28-33，2004.