システム創成工学専攻
情報システムコース
Department of Systems Innovation
Advanced Information System Course

令和元年度 専攻科特別研究論文

バスシステムの構築を容易にする組み合わせ型フレームワークの提案

An Assemblable Framework for Facilitating Bus System Development

指導教員名　上野　秀剛 准教授
論文提出者名　田谷　瑛悟

独立行政法人 国立高等専門学校機構
奈良工業高等専門学校 専攻科
National Institute of Technology, Nara College
Faculty of Advanced Engineering

バスシステムの構築を容易にする組み合わせ型フレームワークの提案

An Assemblable Framework for Facilitating Bus System Development

田谷　瑛悟
Eigo Taya

独立行政法人 国立高等専門学校機構
奈良工業高等専門学校 専攻科 システム創成工学専攻 情報システムコース
大和郡山市矢田町 22 番地（〒 639-1080）
National Institute of Technology, Nara College, Faculty of Advanced Engineering
22 Yata-cho, Yamatokoriyama, Nara 639-1080, Japan

Abstract: Many bus operators and content providers operate systems that provide information such as times and bus routes. However, even if each bus operator wants to provide unique functions or information to increase their competitive power, it is difficult to add them to systems operated by different content providers. Thus, bus operators need to build their systems in that case. On the other hand, bus information is usually provided after it is searched and then assembled, hence most functions provided to users are similar in systems operated by different operators. Therefore, in this research, I propose a framework that assembles common functions in many systems to help develop unique systems. The framework provides 1) basic functions (primitive functions) that are necessary to build functions for providing to users and 2) general functions (integrated functions) of bus systems that are defined as assembled primitive functions. Developers can reduce the time and effort of development other than the development of unique functions by assembling primitive functions or using integrated functions.

Keywords: Software Framework, Bus Transport, Navigation System,

関連業績リスト

1. 田谷 瑛悟, 上野 秀剛 : “バスシステムの構築を容易にする組み合わせ型フレームワークの提案”, 研究報告情報システムと社会環境（IS）, 2019-IS-148, No.4, pp.1-8 , (2019).

目次

図目次

表目次

1. はじめに

全国的にバス利用者が減少し続けており，これに伴い十分な収益を得られないバス路線を廃止・縮小する事例が発生している [1]．この要因の 1 つとして，バス利用者が時刻表や経路といったバスの運行に関する情報（バス情報）を十分に得られないことが指摘されている [2]．時刻表や路線は更新頻度が低い，静的情報ではあるもののバス情報を十分に得られないと，利用者は目的地に移動するための適切な手段か判断できないため利用をあきらめたり，他の移動手段に比べて利便性が高いにもかかわらず気がつかない場合がある．近年では海外観光客の増加に伴い，対象のバスを日常的に利用していない利用者が増加することが見込まれ，静的なバス情報の提供方法については今後も様々な改善が必要と考えられる．

また, バスは電車など他の公共交通機関と比べて渋滞や道路工事等の交通状況や，降雨や積雪等の天候，乗降にかかる時間のように定時運行に影響する要因が多く，遅れが発生しやすい．このような「待ち時間の不確実性」は，出発予定時間付近に利用者がバス停に到着した場合に，既に出発済みかどうか判断することができず不安を感じる原因となっている．そこで「待ち時間の不確実性」に着目し，バスの現在位置や予想到着時刻のような動的な情報を提供するバスロケーションシステムが一部の事業者によって提供されている．一部の研究では，バスロケーションシステムの導入コスト削減を目的とした，市販のスマートフォンを用いたシステムの提案 [2][3] がされている．

バス情報を提供する多くのシステムは各事業者が個別に開発，運用しているのが一般的だが，NAVITIME[*1]や Yahoo!路線情報[*2]のように複数のバス事業者の情報を一括で扱うサービスも複数存在する．これらのサービスはバス情報のみならず電車や飛行機といった他の公共交通機関の情報も同時に提供しており，異なる交通機関の乗り換えを伴う移動を支援する情報を提供している．一方で，バス事業者以外が運営するサービスの場合，個々

[*1] https://www.navitime.co.jp/
[*2] https://transit.yahoo.co.jp/

のバス事業者が提供したい独自の機能や非定型的な情報提供への対応が難しい．たとえば，あるバス事業者が観光客を対象に地域と期間を限定した乗車券を発行する場合，その販売場所や期間，利用方法などの告知を独自に告知する必要がある．このとき，告知の方法や対象を事業者が決定できることが望ましく，バス事業者以外が管理・運用するシステムでは柔軟な対応が難しい．また，各事業者が競争力の強化を目的に独自の機能を提供したい場合，自社でシステムを開発・運用する必要がある．

本研究ではバス事業者などがバス情報を提供するシステム（バスシステム）を独自に開発することを支援するためのフレームワークを提案する．私は多くのバスシステムが提供する機能の共通性に着目する．バスシステムが提供する機能は，各種の静的・動的なバス情報を，想定利用者の目的に合わせて組み合わせて検索・表示している．たとえば経路検索では乗車する停留所と降車する停留所，乗車時刻などの入力を元に，路線情報と時刻情報を組み合わせて検索し，最適な経路を表示する．ユーザに提供する機能（たとえば経路検索）を構築するために必要な基本的な機能（路線検索や時刻検索）をプリミティブ機能と定義し，これらを組み合わせる事で開発者が容易に独自のユーザ提供機能を開発できるようにする．時刻表検索や運賃検索といった多くのバスシステムが提供する一般的な機能については，プリミティブ機能の組み合わせとして事前に定義しフレームワークが提供する．

また，本フレームワークは国土交通省が策定した「静的バス情報フォーマット」と「動的バス情報フォーマット」を用いてバス情報を記録，または外部システムから取得する．これらのフォーマットは 2019 年 3 月に策定され，今後多くのバスシステムでの利用が見込まれるほか，オープンデータとして公開する事業者が増えると考えられる．共通フォーマットで表現されたデータを対象に，基本的な機能を組み合わせる事で必要な機能を容易に実装できる本フレームワークは，バス事業者による独自システムの開発・運用を支援する．

2. バスシステム

2.1 バス情報のフォーマット

日本においてバス情報はバス事業者が独自のフォーマットを策定し，管理している．複数のバス事業者の情報を扱うサービスの事業者（コンテンツプロバイダ）は各バス事業者と契約を交わし，各事業者のバス情報を取得することで，サービスを運用している．このとき，事業者によってフォーマットが異なるためコンテンツプロバイダにはデータフォーマットの変換・管理に大きな負担がかかり，整備のコストに見合わない地方のバス情報は整備されておらず，バス利用者に十分な情報提供ができていなかった [4]．そこで，バス情報の提供を促進することを目的とした「静的バス情報フォーマット」が，国土交通省により策定された [5]．静的バス情報フォーマットは，欧米を中心に公共交通データのデファクトスタンダードとなっている GTFS(General Transit Feed Specification) に日本のバス事業者独自の情報を加えたものである．

「静的バス情報フォーマット」の基となっている GTFS では静的な交通データ以外にも，GTFS Realtime と呼ばれる運行情報や車両の位置情報といった動的な交通データのフォーマットが策定されている．動的な交通データは，乗車予定だった車両が既に出発済みかわからない，事故や悪天候によって運行休止しているかわからないといった利用に対する不安を軽減し，交通機関の利便性向上に寄与する．日本では動的なバス情報に関するフォーマットが策定されておらず，バス事業者によって動的な情報の種類が異なり，複数のバス事業者の情報を統合したサービスの運用が困難であった．そこで 2019 年に GTFS Realtime を動的なバス情報の標準的なフォーマットとして，「動的バス情報フォーマット」が策定された [6]．

2.2　静的なバス情報

静的バス情報フォーマットを表 2.1 に示す．フォーマットは 17 の CSV ファイルによって定義される．なお，ファイル名の後ろに (_jp) と記載されているものは，日本のバス事業者独自の項目を表す．

静的なバス情報はバス事業者が設置・管理している停留所や各停留所をどのような順で通過するか定めた経路，乗車・降車停留所に応じた運賃などが記載されている．また，バスは事業者や経路によって運賃を前払いするか後払いするか異なるため，支払いタイミングについても記載されている．一般に，静的なバス情報は年に数回程度行われる時刻表や運賃の改定や，停留所名称の変更などが行われるのみで，頻繁な変更は行われない．

2.3　動的なバス情報

公共交通の動的な情報は乗車前の情報 (pre-trip information) と，乗車中の情報 (onbording information) に分類される [7]．乗車前の情報には車両の位置情報や運行情報といったバスの運行に関する情報が含まれ，動的バス情報フォーマット（GTFS Realtime）上で下記のように定義されている．

- TripUpdate(ルート最新情報)：遅延，発着時刻予測，通過
- VehiclePosition(車両位置情報)：車両の緯度・軽度，接近情報，混雑度
- Alert(運行情報)：運行情報の概要，影響（運休，迂回等），原因（天候，事故）

乗車前の情報は，各バスの車載器から定期的に送信される，各運行車両の位置情報や正常なダイヤと比較した遅延時間が記載される．また，長時間運行できない事態が発生した時に，事態の原因や運行への影響なども記載される．乗車前の情報は記載される運行情報と実際の運行情報との誤差が大きいと，バス利用者が乗り遅れたり利用に不安を感じてしまうため，更新間隔は 30 秒以下であることが推奨されている．

一方で動的バス情報フォーマットには乗車中の情報が定義されていない．乗車中の情報には，「正しいバスに乗車できているか」，「降車までの時間はどれくらいか」といった個々の利用者の状況に合わせたバス情報の表示が求められる．たとえば，「正しいバスに乗車できているか」を判断するためには，その地域で運行中のバス路線を把握した上で，車内の行き先表示器や通過した停留所名から乗車中のバスを推測し，比較する必要がある．特

表 2.1: 静的バス情報フォーマット

ファイル名	項目	説明
agency.txt	事業者情報	事業者の名称，HP の url，連絡先
agency_jp.txt	事業者追加情報	運輸行政への申請に必要な事業者の情報
stops.txt	停留所情報	停留所と標柱の名称や地理情報
routes.txt	経路情報	バスの運行経路の略称や正式名称
routes_jp.txt	経路追加情報	運行経路の出発地，目的地，経由地の名称
trips.txt	便情報	各路線に属する運行便の情報
office_jp.txt	営業所情報	便を運行する営業所の情報
stop_times.txt	通過時刻情報	便ごとの各停留所の通過時刻情報
calendar.txt	運行区分情報	平日や土日といった曜日に対する運行情報
calendar_dates.txt	運行日情報	祝日や臨時運行と行った日付に対する運行情報
fare_attributes.txt	運賃属性情報	支払いタイミングや運賃の情報
fare_rules.txt	運賃定義情報	均一制や対距離制といった細かな運賃計算の定義
shapes.txt	描画情報	地図上における標柱間の経路情報
frequencies.txt	運行間隔情報	一定で運行する際に定義する運行情報
transfer.txt	乗換情報	乗り換え地点となる標柱情報
feed_info.txt	提供情報	データを公開する組織情報やデータの有効期限
translation.txt	翻訳情報	名称に対するよみがなや英語の情報

に，利用状況を適切に把握することはバスの利用経験が少ない人や事前知識の少ない観光客にとって困難であり，バスシステムは乗車中の情報を提供し，このようなバス利用者を支援する必要がある．しかし，個々の利用者の状況を把握し乗車中の情報を提供するバスシステムは少ない．一部の研究では乗車中の情報を提供するバスシステムが提案 [8] されているが，乗車中の情報にどんな情報が必要か整理されていない．今後，乗車中の情報についてフォーマットが策定されることが考えられ，提案フレームワークを対応させることについては今後の研究課題である．

2.4　バスシステムの機能

バス利用者の増加や導入コスト削減などを目的に，バスシステムが複数提案されている．Ferris らと Megalingam らは，バス車両に GPS 車載器を取り付けて，各運行車両の位置情報や正常なダイヤと比較した遅延時間といった動的なバス情報を提供するバスロケーションシステムを提案している [9]，[10]．また，鴫原らと伊藤らは GPS 車載器の代わりにスマートフォンを利用することで，バスロケーションシステムの導入コストを削減している [2], [3].

Diego らは，上記の動的なバス情報に加えて観光地への移動方法も合わせて提供できるシステムを提案している [11]．このシステムにより観光客のような日常的にバスを利用しない人がバスを利用しやすくなることが示されている．

Foell らは，「正しいバスに乗車できているか」，「降車までの時間はどれくらいか」といった個々の利用者の状況に合わせたバス情報を提供するシステムを提案している [8]．また，同著者は上記のバス情報に加えて，バス車両の混雑度を推測し，より混雑していないバス経路を表示できるシステムを提案している [12].

以上の研究のように，様々なバスシステムが提案されているが，バスシステムを開発するコストについて議論されていない．多くのバス事業者では十分な収益を得られておらず [13]，個々の事業者がバスシステムを開発することは困難である．そのため，事業者によるバスシステム開発を支援する必要がある．

バスシステムは各事業者が個別に開発，運用しているのが一般的だが，コンテンツプロバイダが運用するような複数のバス事業者の情報を一括で扱うサービスも存在する．一方で，バス事業者以外が運営するサービスの場合，個々のバス事業者が提供したい独自の機能や非定型的な情報提供への対応が難しい．各事業者が競争力の強化を目的に独自の機能を提供する場合，自社でシステムを開発・運用する必要がある．

そこで私はバスシステムで共通する機能をまとめ，開発を支援するフレームワークを提案する．バスシステムがユーザに提供する機能はバス情報を組み合わせて検索・表示している．例えば，ある停留所の時刻表の表示には停留所，経路，便，通過時刻，運行日情報を組み合わせ，運行系統図の表示は停留所，経路，便，描画情報を組み合わせて検索・表示することで実現されている．フレームワークはユーザに提供する機能を構成するために必要な基本的な機能をプリミティブ機能として定義し，これらを組み合わせることでユーザ

に提供する機能を容易に開発可能にする．また，様々な事業者によってバスシステムが運用されているが，停留所の時刻表の表示や運行系統図の表示といったユーザに提供する機能は大部分が類似している．フレームワークは多くのバスシステムが提供する機能をプリミティブ機能の組み合わせとして定義し，独自機能以外にかかる開発の手間を削減する．

なお，プリミティブ機能は 2.2 節と 2.3 節で説明した静的・動的バス情報フォーマットの項目（停留所情報，経路情報）毎に定義する．バス情報フォーマットを用いてフレームワークを定義することで，異なるフォーマットを持つ複数のバス事業者に対応することが可能となる．

3. 提案フレームワーク

3.1 概要

バスシステムに共通する機能の提供と，独自機能の開発容易化のために，提案フレームワークは以下の設計方針に従って作成されている．

1. プリミティブ機能の提供
   フレームワークはユーザに提供する機能を構成するための，基本的な機能（プリミティブ機能）を提供する．バスシステムがユーザに提供する機能の多くは，ユーザの特定の目的に合わせてバス情報を検索，統合することで実装される．例えば「時刻表の表示」機能はあるバス停におけるバスの通過時刻の一覧を検索することで実装される．また，「乗車経路検索」機能は乗車するバス停の名前と降車するバス停の名前と同じ名前をもつバス停を検索し，それらの属する路線情報や乗り換え可能なバス停，各バス停におけるバス到着予定時刻，運賃表などの検索結果を組み合わせることで実装される．フレームワークは個々のバス情報を検索する機能を提供し，バスシステムの開発者がこれらを組み合わせることで容易に必要なユーザ機能を開発可能にする．
2. 統合機能の提供
   異なる事業者が提供するバスシステムであっても，多くのシステムが提供している一般的な機能が存在する．たとえば，ある停留所の時刻表検索は多くのバスシステムで提供されており，新規に開発するシステムにおいても開発する可能性が高い．提案フレームワークはこのような一般的な機能を，複数のプリミティブ機能を組み合わせた「統合機能」として事前に定義し，開発者に提供することで独自機能以外の開発にかかる手間を削減する．
3. 外部システムからのバス情報の取得
   プリミティブ機能の動作に必要な静的・動的バス情報は，外部にあるデータベース

や WebAPI などから取得する．1 章で述べた通り，静的・動的バス情報フォーマットが国土交通省により 2019 年 3 月に策定された．今後，多くのバスシステムにおいて同フォーマットが用いられるとともに，オープンデータとして公開されたり，Web API を経由した利用が可能になるバス事業者が増加すると見込まれる．本フレームワークでは内部にデータを持たせずに既存の外部システムから取得することで，データ更新などの管理を追加で発生させない．フレームワークは外部システムとの連携機能を容易に実装するためのクラス群を開発者に提供する．新たに開発される外部システムとの連携機能はプリミティブ機能の 1 つとして実装され，他のプリミティブ機能と組み合わせることで統合機能の作成に利用される．

4. 外部システムによる関連情報の取得
   一部のバスシステムには，電車などバスとの乗り継ぎで使用する移動手段についての情報をバス情報と組み合わせて提供する機能が存在する．例えば，周辺の土地情報に詳しくないバス利用者は，目的地までの行き方がわからないため，出発地からバス停，あるいはバス停から目的地への徒歩経路は需要が高い．フレームワークはバス以外の交通機関のシステムや地図検索，徒歩経路検索といった外部システムに接続する外部連携機能を提供する．これにより徒歩経路作成や他交通機関の乗換検索といった新たなプリミティブ機能を定義できる．

3.2 アーキテクチャ

図 3.1 に提案フレームワークを用いて開発するバスシステムのクラス図を示す．図中の灰色は提案フレームワークが提供する機能の範囲を示す．フレームワークは主に 2 つのパッケージ 1) プリミティブと，2) 統合機能からなる．

プリミティブパッケージ（図の中段）にはバス情報システムの開発に必要な基本機能と，基本機能によって検索されるバス情報を格納するクラスが複数含まれる．なお，図中には紙面の都合上，主要なクラスのみを記載している．プリミティブパッケージ内のクラスは抽象クラスとして定義され，実際に使用される各事業者のバス情報 API やデータベースへのアクセスは子クラスが行う．プリミティブパッケージとその実装クラスは Abstract Factory パターンを構成しており，異なるバス会社のデータへのアクセスをサポートするとともに，フレームワーク内でのデータ表現と検索処理を統一する．また，開発者はプリミティブパッケージを拡張することで，新たな外部サービス（例えば電車の運行状態を提

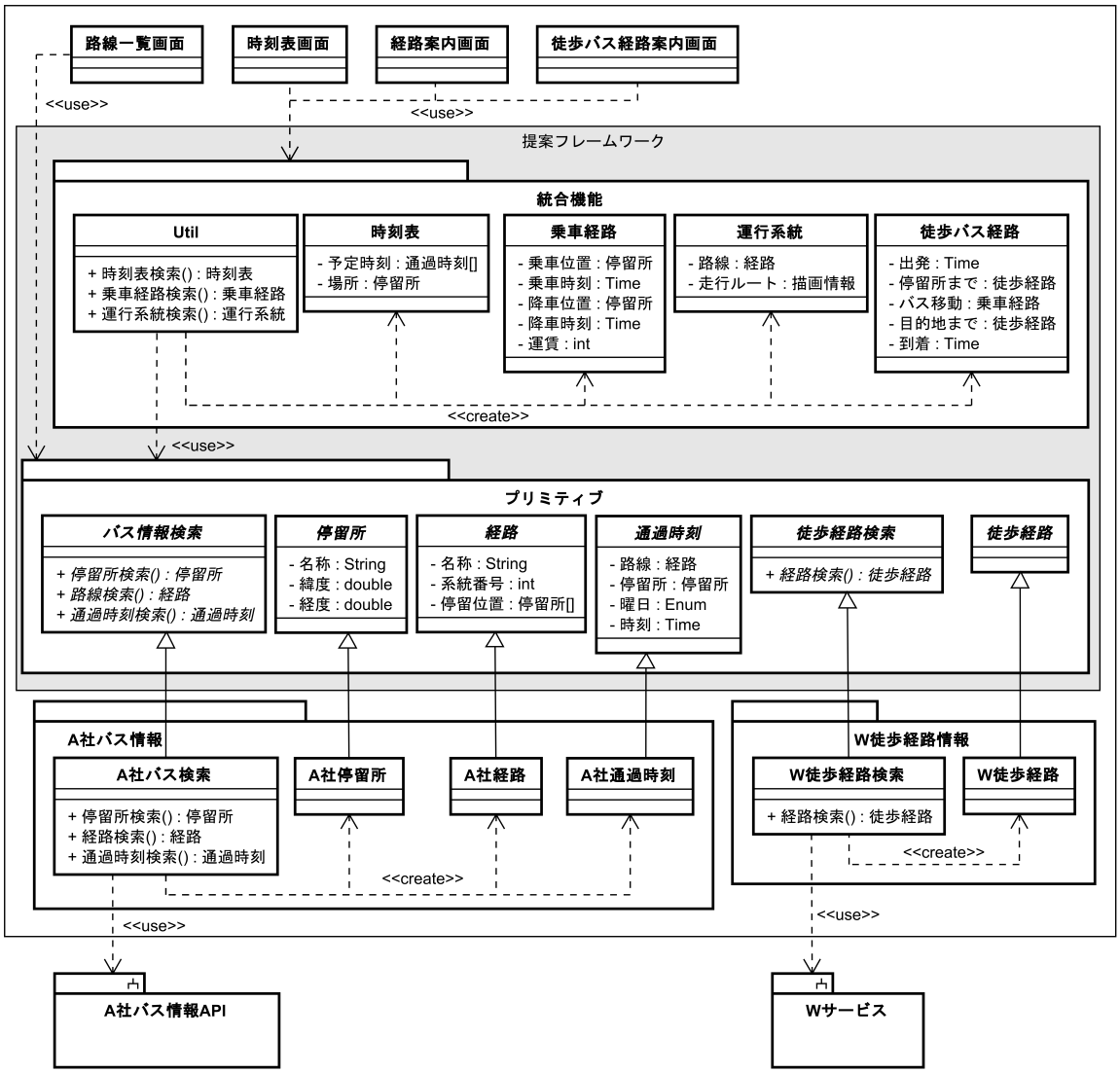

図 3.1: 提案フレームワークを用いたシステムのクラス図

供する API）を用いた機能を開発できる．

統合機能パッケージ（図の上段）にはバスシステムのユーザ機能に対応する機能群と，ユーザ機能によって検索されるバス情報を格納するクラスが複数含まれる．統合機能はプリミティブ機能を組み合わせることで実現され，一般的なバスシステムで必要と思われる機能については提案フレームワーク内であらかじめ定義する．開発者は定義済みの統合機能を利用することで，一般的な機能を備えた独自のバスシステムを容易に開発できる．ま

た，開発者はプリミティブ機能や既存の統合機能を利用することで独自の統合機能を容易に開発できる．

3.3 プリミティブ機能

プリミティブ機能はユーザに提供する機能を構成するための，個々のバス情報を検索する機能である．フレームワークが提供するプリミティブ機能の一覧を表 3.1 に示す．プリミティブ機能の多くは静的なバス情報フォーマットで定義された，バスシステムの運用に必要と思われる項目について，検索機能を提供する．また，動的なバス情報フォーマットで定義された項目の一部と，地図上への経路や停留所の表示，2 地点間の徒歩経路を検索する機能についても多くのバスシステムにおいて有用と思われるため提供する．徒歩経路の検索機能については，Google Maps Platform の Directions API を利用するためのクラス群（図 3.1 の W 徒歩経路情報パッケージ内のクラスに相当）を作成し，フレームワークに含めている．

プリミティブ機能は検索を行うためのメソッド群を持つクラス（図 3.1 のバス情報検索クラスや徒歩経路検索クラス）と，各メソッドの戻り値となるバス情報を格納するクラス群（図 3.1 の停留所や路線，徒歩経路など）から構成される．開発者はフレームワークが提供する上記のクラスを親クラスとする，情報源の API やデータベースにアクセスするための専用クラスを開発して利用する．情報源ごとにアクセス方法や情報の取得方法が異なるためそれぞれ個別に開発する必要があるが，標準フォーマットに従って情報が提供されているバス情報の場合，親クラスで用意されているメソッド群を利用することで開発を容易にする．プリミティブ機能によって取得された情報はフレームワークで定義された，バス情報を格納するためのクラスに格納される．フレームワークが用意していないプリミティブ機能については開発者が個別にプリミティブパッケージに機能を追加することで，既存のプリミティブ機能と組み合わせた開発が可能になる．

3.4 統合機能

統合機能はプリミティブ機能を組み合わせて作成される，ユーザに提供する機能である．フレームワークが提供する統合機能の一覧を表 3.2 に示す．3 つの統合機能はいずれも一般的なバスシステムで提供される機能である．例えば「時刻表検索」は指定の停留所に設定された時刻表を検索する．時刻表の作成は表 3.2 に示す 5 つのプリミティブ機能で

表 3.1: フレームワークが提供するプリミティブ機能

機能	説明
事業者検索	静的バス情報フォーマットで定義された情報を検索する
停留所検索	
経路検索	
便検索	
営業所検索	
通過時刻検索	
運行区分検索	
運行日検索	
運賃属性検索	
運賃定義検索	
描画情報検索	
運行間隔検索	
乗換方法検索	
提供者検索	
翻訳情報検索	
ルート最新情報検索	動的バス情報フォーマットで定義された情報を検索する
車両位置検索	
運行情報検索	
地図描画	地図上に経路や停留所などを表示する
徒歩経路検索	2 地点間の徒歩経路と移動時間を取得する

得られるバス情報を組み合わせることで実現され，時刻表クラスに必要な情報が格納される．

統合機能は検索を行うためのメソッド群を持つクラス（図 3.1 の Util クラス）と，各メソッドの戻り値となるバス情報を格納するクラス群（図 3.1 の時刻表や乗車経路，徒歩バス経路）から構成される．開発者は開発するシステムの各ページにおいて，統合機能を用いて必要な情報（例えば時刻表）を検索し，表示画面を作成する．開発者はプリミティブ機能で得られる情報を利用して（統合機能を経由せずに）表示画面を作成したり，プリミ

表 3.2: フレームワークが提供する統合機能

機能	使用するプリミティブ機能	説明
時刻表検索	停留所検索，経路検索，便検索，通過時刻検索，運行日検索	ある停留所の時刻表を検索する
乗車経路検索	停留所検索，経路検索，便検索，通過時刻検索，運行日検索，運賃属性検索，運賃定義検索	任意の 2 停留所間の移動経路と運賃を，時刻を指定して検索する
運行系統図検索	停留所検索，経路検索，便検索，描画情報検索，地図描画	地図上に任意の路線に含まれる停留所と経路を表示する
徒歩バス経路検索	停留所検索，経路検索，便検索，通過時刻検索，運行日検索，運賃属性検索，運賃定義検索，描画情報検索，地図描画，徒歩経路検索	地図上に任意の 2 地点を結ぶ，徒歩とバス移動を組み合わせた経路を表示する

ティブ機能を組み合わせて独自の統合機能を作成することも可能である．

4. モデルケース

提案フレームワークを用いたシステム構築のモデルケースによって，(a) バスシステムの一般的な機能を容易に開発できるか，(b) 独自の機能・情報を追加できるか確認する．

4.1 観光案内システム

観光地を訪れた観光客は一般に観光地の地理を詳細には把握しておらず，観光スポットまでの移動に困難を感じることがある [8]．このような観光客を対象に，周辺の観光スポットや移動方法を提示することで観光行動を促進する観光案内所が各自治体や観光協会，バス事業者などによって運営されている．しかし，バス事業者が運営する観光案内所職員の話によると，SNS や口コミサイトの普及によって定番ではない観光スポットや店舗への行き方を観光客に尋ねられる場合があり，職員が対応できない場合がある．本モデルケースでは，観光案内所の職員が使用する，徒歩とバスを組み合わせた経路を検索・印刷する機能を持つ観光案内システムの構築を行う．システムは任意の出発地と目的地を入力すると以下の経路を生成し，地図上に表示する．システムが出力する経路を以下に示す．

1. 出発地から最寄りの停留所までの徒歩経路
2. 停留所から目的地最寄りの停留所までのバス経路
3. 停留所から目的地までの徒歩経路

また，システムは各地点の出発時間と移動に要する時間，バスの発車時刻，運賃を表示する．案内所の職員は観光客に尋ねられた目的地の住所，あるいは名称を入力し，経路を検索する．そして，職員は必要に応じて検索結果を印刷して観光客に手渡す．

観光案内システムが出力する徒歩経路，バス経路，発着時刻，運賃情報は，多くのバスシステムがユーザに提供する情報である．本モデルケースではフレームワークを利用することで，これらの情報を検索する機能を実装できるか確認する．これにより観点 (a) バス

システムの一般的な機能を容易に開発できるかを確認する．本システムの実装には以下のバス情報を検索し，適切に組み合わせる必要がある．

- 任意の地点（出発地や目的地）の近くにある停留所
- 各停留所の所属する運行系統（路線）
- 2 つの停留所が同じ運行系統に属していない場合，乗り換え可能な停留所
- 現在地から停留所まで，および停留所から目的地までの徒歩経路，および移動時間
- 現在時刻と徒歩での移動時間を考慮した，乗車可能なバスの出発時刻，到着時刻，および運賃
- 地図上で表示するバス路線の描画情報

フレームワークはユーザ機能を構成する基本的な機能（プリミティブ機能）を提供し，現状の実装では標準フォーマットで定義された項目についての検索機能，および，GoogleMap に代表される地図サービスが提供する地図上への表示，入力された出発地と目的地間の徒歩経路を検索する機能を提供している．上記のバス情報はいずれもフレームワークの提供するプリミティブ機能で取得可能である．つまり，開発者はバスシステムのユーザ機能に必要な個々の情報機能を実装することなく，フレームワークの機能を組み合わせることでユーザ機能を実装できる．また，フレームワークはバスシステムが提供するユーザ機能として，プリミティブ機能の組み合わせである統合機能を実装済みである．すなわち，多くのバスシステムにとって実装が必要と思われるユーザ機能を，開発者は実装することなく開発するシステムに組み込むことができる．上記のバス情報のうち，各停留所からの出発時刻，移動時間，運賃は統合機能の「乗車経路検索」によって，出発地から目的地までのバス・徒歩経路は「徒歩バス経路検索」によっても取得可能である．開発者は開発状況に応じて，プリミティブ機能を組み合わせるか，統合機能を利用するか選択し，機能を実装できる．たとえば，バス事業者が他にバスシステムを運用していない状況で，観光案内システムを構築する場合を考える．この場合，開発者は統合機能の「徒歩バス経路検索」を用いることで，ユーザ機能を実装することなくシステムに組み込める．一方で，バス事業者が既に「任意の地点の近くにある停留所」,「各停留所の所属する運行系統」,「乗り換え可能な停留所」の情報を組合わせて検索できる統合機能を既に構築していた場合を考える．この場合，システムの開発者はフレームワークのプリミティブ機能と既に構築済みの統合機能を組み合わせることで，システムを構築できる．

フレームワーク利用による実装の削減量の例として,「乗車経路検索」をフレームワー

クを利用せずに実装したプログラムを図 4.1 に，利用して実装したプログラムを図 4.2 に示す．プログラムはいずれもプログラミング言語「Ruby」を用いて実装され，2.2 節のフォーマットに沿ったバス情報を NoSQL データベース「MongoDB」から取得している．バスシステムは Web アプリケーションとして運用されていることが多く，Web アプリケーションの開発にはプログラミング言語「Ruby」が用いられることが多い．そこで乗車経路検索の実装には，プログラミング言語の「Ruby」を用いた．また，乗車経路検索に必要なバス情報を保存するデータベースは，Web アプリケーションのデータベースとして用いられることの多い「MongoDB」を用いた．乗車経路検索は，任意の 2 停留所間の移動経路と運賃を，時刻を指定して検索する．引数として出発停留所名 (origin_name)，到着停留所名 (destination_name)，指定日時 (target_date)，バス情報が保存されているデータベースのホスト名あるいは IP アドレス (host)，ポート番号 (port)，乗車経路検索に必要なバス情報（停留所，経路，便，通過時刻，運行区分，運行日，運賃属性，運賃定義）を保存しているデータベースの名称 (db_name) を渡す．また，返り値として出発停留所，出発時刻，到着停留所，到着時刻，料金の情報を 1 要素とする，複数の要素から構成される配列 (result) を返す．

フレームワークを利用しないプログラム（図 4.1）は，はじめにデータベースへの接続に必要なパラメータを定義している (6〜10 行目)．ここで定義したパラメータは，停留所や料金情報といったバス情報をデータベースから取得する際に用いる．次に，運行区分 (calendar)・運行日 (calendar_date) 情報から，指定日 (target_date) に運行している service_id を取得している (13〜22 行目)．service_id は，指定日に運行している便情報を取得するための条件 (50 行目) に用いる．なお，16・19 行目では find メソッドを用いて条件に合致するドキュメントの集合を取得している．条件は，16 行目では運行区分情報が表す運行開始日から運行終了日までの期間に指定日 (target_date) があるか，19 行目では運行日情報が表す日付と指定日が同じか，と指定している．find メソッドでの条件指定は $find(: field\_name => value)$ のように記述し，比較演算する場合は $find(: start\_date => "\$lte" => target\_date)$ のようにオプションを記述する．ここで，$\$lte$ は指定した value 以下のドキュメントを取得し，$\$gte$ は指定した value 以上のドキュメントを取得することを意味している．また，17 行目では if 修飾子と呼ばれる Ruby 特有の記法を利用している．if 修飾子は if より右辺の条件 ($c[weekdays[wday]] == 1$) が true のとき，左辺の式 ($service\_ids.push(c[: service\_id]$) を実行し，より記述をコンパクトにしたいときに用いられる．次に，プログラムは出発停留所（25〜31 行目），到

着停留所 (34〜40 行目) の停留所情報 (stop) を取得した後，通過時刻情報のフィールド「stop_id」で紐付けられた通過時刻情報 (stop_times) を取得している．通過時刻情報には返り値に必要な出発・到着時刻が記述されており，返り値 result を生成する際に参照される．次に，プログラムは料金，出発停留所の発車時刻，到着停留所の到着時刻の情報を取得し，返り値 (result) を生成している (42〜69 行目)．25〜31 行目，および 34〜40 行目で取得した通過時刻情報は，出発・到着停留所のそれぞれで便[*1]の対応がされていない．そこで出発・到着停留所それぞれの通過時刻情報に対して繰り返し処理を記述し，同一の便にあるか判定している．もし同一便にある場合，料金と通過時刻情報を取得し，返り値 result に格納している．最後に，プログラムは返り値となる配列 result を出発時刻の昇順で並び替え，返している (71 行目)．

フレームワークを利用したプログラム（図 4.2）はフレームワークのメソッドである trip_route を呼び出し，フレームワークを利用しないプログラム（図 4.1）と同じ検索結果を変数 result に保存している．データベースへの接続や停留所情報の取得といった処理はフレームワーク内で処理するため，図 4.2 のプログラムでは記述する必要がない．

[*1] 乗客が連続して乗車可能な 1 回の運行

```ruby
# -*- coding: utf-8 -*-
require 'rubygems'; require 'mongo'; require 'time'

def trip_route(origin_name, destination_name, target_date, host, port, db_name)
    # データベースと接続
    client = Mongo::Client.new([ host + ':' + port ], :database => db_name)
    stopDB = client[:stops]; stop_timesDB = client[:stop_times];
    routeDB = client[:routes]; tripDB = client[:trips]
    calendarDB = client[:calendar]; calendar_datesDB = client[:calendar_dates]
    fare_rulesDB = client[:fare_rules]; fare_attributesDB = client[:fare_attributes]

    # 指定日に運行している service_id を取得
    service_ids = []
    wday = Time.parse(target_date.to_s).wday
    weekdays = ["sunday","monday","tuesday","wednesday","thursday","friday","saturday"]
    for c in calendarDB.find(:start_date => {"$lte" => target_date}, :end_date => {"$gte" =>
          target_date})
        service_ids.push(c[:service_id]) if c[weekdays[wday]] == 1
    end
    for c in calendar_datesDB.find(:date => target_date)
        service_ids.delete(c[:service_id]) if c[:exception_type] == 2
        service_ids.push(c[:service_id]) if c[:exception_type] == 1
    end

    # 出発停留所の情報を取得
    o_stop = stopDB.find(:stop_name => origin_name)
    o_stop_times = []
    for stop in o_stop
        for stop_time in stop_timesDB.find(:stop_id => stop[:stop_id])
            o_stop_times.push(stop_time)
        end
    end

    # 到着停留所の情報を取得
    d_stop = stopDB.find(:stop_name => destination_name)
    d_stop_times = []
    for stop in d_stop
        for stop_time in stop_timesDB.find(:stop_id => stop[:stop_id])
            d_stop_times.push(stop_time)
        end
    end

    result = []
    for o in o_stop_times
        for d in d_stop_times
            # 同一便の通過時刻情報を参照
            next if o[:trip_id] != d[:trip_id]

            # 料金情報取得に必要な便情報を取得
            trip = []
            tripDB.find(:trip_id => o[:trip_id], :service_id => {'$in' => service_ids}).each{ |
                r| trip = r }

            # 料金情報を取得
            next if trip.empty?
            fare_rule = []
            fare_rulesDB.find(:route_id => trip[:route_id], :origin_id => o[:stop_id],
                :destination_id => d[:stop_id]).each{ |f| fare_rule = f }

            next if fare_rule.empty?
            fare_attribute = []
            fare_attributesDB.find(:fare_id => fare_rule[:fare_id]).each{ |f| fare_attribute =
                f }

            # 検索結果を配列に格納
            result.push(
                departure_stop: origin_name, departure_time: o[:departure_time],
                arrival_stop: destination_name, arrival_time: d[:arrival_time],
                fare: fare_attribute[:price]
            )
        end
    end

    return result.sort! {|a, b| a[:departure_time] <=> b[:departure_time]}
end
```

図 4.1: フレームワークを利用しない乗車経路検索プログラム

```
1 # -*- coding: utf-8 -*-
2 require ' ./busframework'
3
4 framework = BusFramework.new
5 result = framework.trip_route(origin_name, destination_name, target_date, host, port, db_name)
```

図 4.2: フレームワークを利用した乗車経路検索プログラム

乗車経路検索プログラムの SLOC(Source Lines of Code：コメントや空行を含まないコード行数) は，フレームワークを利用しないプログラム (図 4.1) は 53 行，フレームワークを利用するプログラム (図 4.2) は 3 行であり，フレームワークを利用するプログラム (図 4.2) の方が SLOC は小さい．このことからフレームワークを利用することで，乗車経路検索を実装する手間を削減できていることがわかる．そのため，観光案内システムに必要な実装のうち，各停留所からの出発時刻，移動時間，運賃の取得は「乗車経路検索」で処理できる．また，バス時刻・運賃を検索できる機能を提供する Web システムを実装する場合にも，各停留所からの出発時刻，移動時間，運賃を取得する必要がある．これらの情報取得も観光案内システムと同様にフレームワークの「乗車経路検索」で取得できる．

このようにバスシステムの機能を実装する際，フレームワークを利用することでバス情報を検索する処理やバス情報を組み合わせる処理を実装する手間を削減できる．そのため，本システムの開発者は外部システムからバス情報を取得するためのクラスと，フレームワークの機能が出力する結果を表示する画面を作成するだけよい．以上の点から，提案フレームワークはバスシステムの一般的な機能を容易に開発できるといえる．

4.2 デジタルサイネージ

多くの停留所には時刻表や案内板といった紙媒体の掲示物が設置されている．これら紙媒体の掲示物は，バス情報が更新される度に全掲示物の情報を更新する必要があった．加えて，掲示情報の多言語対応や，周辺観光地の情報掲載，遅延情報のような動的な情報に対する需要が大きくなっており，ディスプレイやプロジェクタを用いたデジタルサイネージが導入されてきている．本モデルケースでは 1) 多言語による時刻表の表示，2) リアルタイムな運行状況の提示，3) 周辺観光スポットの案内が可能なデジタルサイネージシステムの構築を行う．

図 4.3 にフレームワークを用いたデジタルサイネージシステムのクラス図を示す．灰色

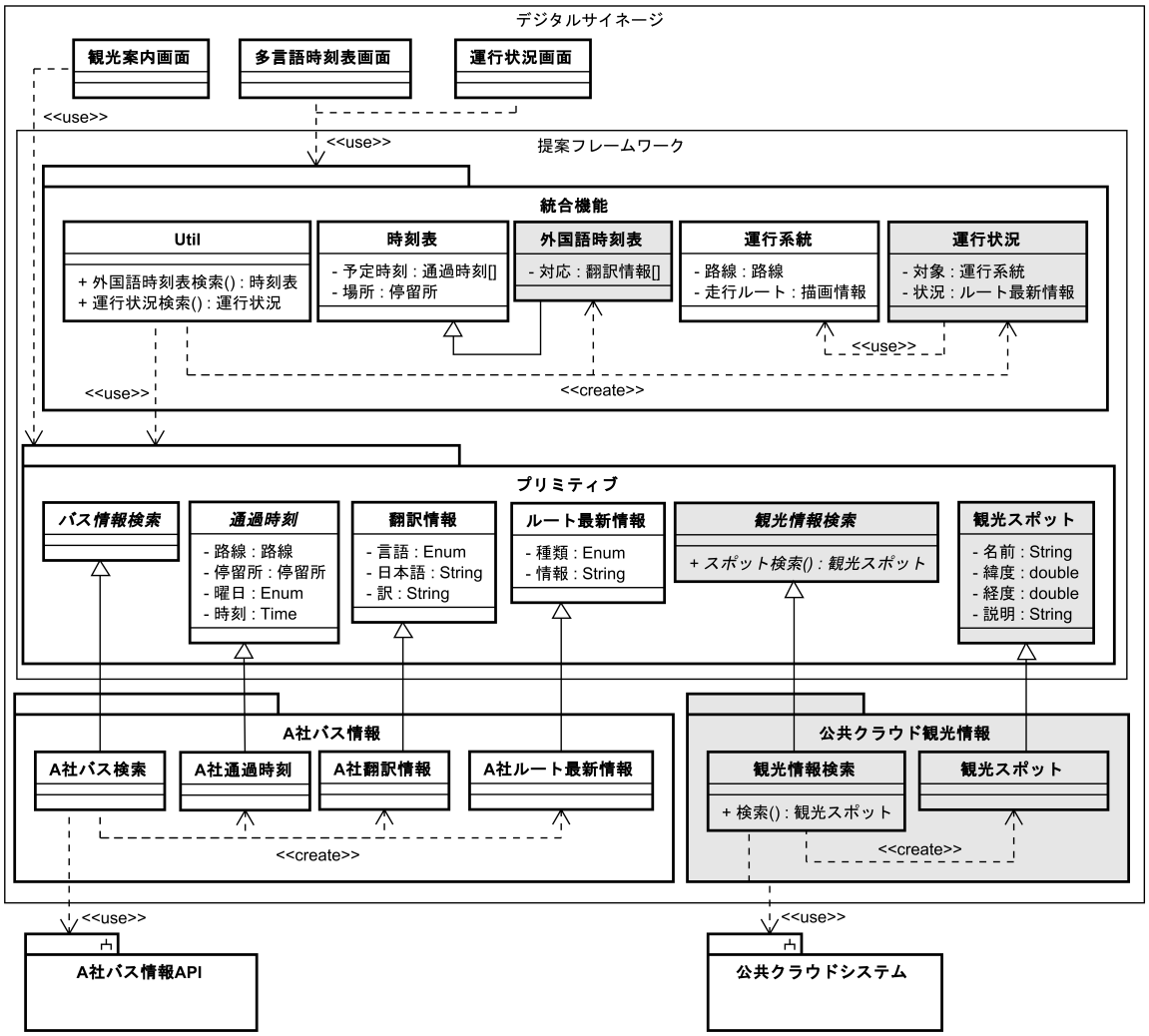

図 4.3: フレームワークを用いたデジタルサイネージのクラス図

の部分はシステムの構築にあたりフレームワークに追加が必要な部分を示す．多言語による時刻表の表示とリアルタイムの運行状況の表示はいずれもフレームワークが提供するプリミティブ機能を組み合わせることで作成できる．図の外国語時刻表は既存の時刻表クラスを拡張し，プリミティブ機能の 1 つである翻訳情報検索によって他言語での表記情報を組み合わせることで実現する．運行情報は既存の統合機能である運行系統に，動的情報の 1 つであるルート最新情報を加えることで実現できる．

本システムを構築するためには，バス情報を提供する API に加えて，観光情報を提供

する API が必要である．このモデルケースでは総務省が提供している，全国の観光情報のオープンデータを提供する公共クラウドシステム[*2]を利用する．開発者は公共クラウドシステムに接続するための処理を公共クラウド観光情報パッケージ（図右下）に作成する必要がある．これらの機能を用いて，開発者は周辺の観光スポットを案内する画面を作成できる．以上の点から，提案フレームワークを用いた開発ではフレームワークが提供する機能を使うことで，独自機能の追加が可能であるといえる．

[*2] https://www.chiikinogennki.soumu.go.jp/k-cloud-api/

5. おわりに

本稿ではバス事業者などがバスシステムを独自に開発することを支援するためのフレームワークを提案した．フレームワークはユーザに提供する機能を構成するためのプリミティブ機能と，プリミティブ機能を組み合わせた統合機能を提供し，独自機能以外の開発にかかる手間を削減する．また，システム構築のモデルケースによって提案フレームワークを評価し，提案フレームワークがシステム構築に有用である可能性を示した．

今後の課題として，実装したフレームワークを用いたシステム開発を行い，必要なプリミティブ機能や統合機能の洗い出しが挙げられる．提案フレームワークはプリミティブパッケージにクラスを追加することで新たなプリミティブ機能を追加することが可能であるが，多くのバスシステムの必要とされる機能については，あらかじめ用意されていることが好ましい．加えて，フレームワークが接続する外部システムについて，Google Map Platform に代表される主要なサービスについてはあらかじめフレームワーク上に用意することで開発者の手間をさらに削減することも可能である．

また，近年，国内外を問わず Mobility as a Service(MaaS) への取り組みが推進されている．MaaS は複数種類の移動手段における，検索・予約・決済などのサービスを単一のサービスに統合して提供する概念である．この MaaS の実現にはバスだけでなく，多様な移動手段の情報を一元化して提供するシステムが必要となる．フレームワークには，バス以外の移動手段の情報を扱うプリミティブ機能を新たに定義できる．新たに定義されるプリミティブ機能を含め，それらの組み合わせ方を検討することで，多様な移動手段の情報を提供するシステムの構築を容易にできると考えられる．

謝辞

本論文の執筆，および研究をすすめるにあたり，様々な方々に御協力を賜りました．ここに謝意を添えて御名前を記させていただきます．

指導教員である上野 秀剛准教授には論文の添削や発表練習，ミーティングなど，様々な場面でご指導いただきました．心より感謝を申し上げます．

岡村 真吾准教授には査読において研究を様々なご指摘をいただきました．心より感謝を申し上げます．

松村 寿枝教授，山口 賢一准教授，市川 嘉裕助教には研究力向上セミナーにおいて様々な視点から御指摘とご助言をいただきました．心より感謝を申し上げます．

上野研究室の皆様，及び専攻科生の同期の方々には，日々の議論を通じて大変参考になる意見をいただきました．また，私の他愛もない雑談に付き合っていただきました．おかげで研究室で有意義で楽しい時間を過ごすことが出来ました．ありがとうございました．

参考文献

[1] 国土交通省: “地域公共交通に関する最近の動向等”, http://www.mlit.go.jp/common/001134509.pdf, (2017.12.12).

[2] 鴫原 育子, 山田 稔, 齋藤 修, 兼子 恭平: “利用者位置から検索するバスナビゲーションシステムに関する研究”, 土木学会論文誌 (土木情報学), Vol.70, No.2, pp.I_293-I_302, (2014).

[3] 伊藤昌毅, 川村尚生, 菅原一孔: “スマートフォンを利用したバスロケーションシステムの開発”, 電子情報通信学会和文論文誌 D, Vol.J96-D, No.10, pp.2327-2339, (2013).

[4] 伊藤昌毅, 瀬崎薫: “日本における公共交通オープンデータの現状と展望”, 第 55 回土木計画学研究発表会・講演集, (2017).

[5] 国土交通省: “静的バス情報フォーマット (GTFS-JP) 仕様書（第 2 版）”, http://www.mlit.go.jp/common/001282276.pdf (2019.03.28).

[6] 国土交通省: “動的バス情報フォーマット（GTFS リアルタイム）ガイドライン”, http://www.mlit.go.jp/common/001282183.pdf (2019.03.28).

[7] Dias Camacho T, Foth M, and Rakotonirainy A: “*Pervasive Technology and Public Transport: Opportunities Beyond Telematics*” , IEEE Pervasive Computing, Vol.12, No.1, pp.18-25, (2013).

[8] Foell S, Kortuem G, Rawassizadeh R, Handte M, Iqbal U, and Marrón P: “*Micro-Navigation for Urban Bus Passengers: Using the Internet of Things to Improve the Public Transport Experience*” , In Proc. the First International Conference on IoT in Urban Space, pp.1-6, (2014).

[9] B. Ferris, K. Watkins, and A. Borning, “*OneBusAway: A Transit Traveler Information System*”, in Mobile Computing, Applications, and Services. Springer

Berlin Heidelberg, pp. 92–106, (2010)

[10] R. K. Megalingam, N. Raj, A. L. Soman, L. Prakash, N. Satheesh and D. Vijay, “*Smart, public buses information system*” , International Conference on Communication and Signal Processing, pp. 1343-1347, (2014).

[11] Diego H. B. Zanchett, Jurair R. P. Junior, André F. Monteiro, Diego B. Haddad, Laura S. Assis, “*Collaborative information system to find efficient routes using public transport*”, WebMedia ’19: Proceedings of the 25th Brazillian Symposium on Multimedia and the Web, pp. 473–476 (2019)

[12] M. Handte, S. Foell, S. Wagner, G. Kortuem and P. J. Marrón, “*An Internet-of-Things Enabled Connected Navigation System for Urban Bus Riders*”,IEEE Internet of Things Journal, vol. 3, no. 5, pp. 735-744 (2016).

[13] 国土交通省: “平成３０年度の一般乗合バス事業（保有車両３０両以上）の収支状況について”, `https://www.mlit.go.jp/report/press/content/001318933.pdf`, (2020.01.28).