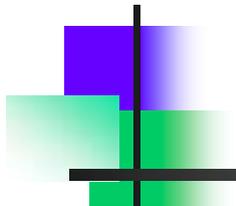


【道路交通情報Webサービスを使った複合Webサービス実証実験成果資料】

A decorative graphic on the left side of the slide, consisting of a vertical black line and a horizontal black line intersecting at a point. To the left of the intersection are two overlapping squares: a purple one on top and a green one on the bottom, both with a gradient effect.

道路交通情報Webサービス

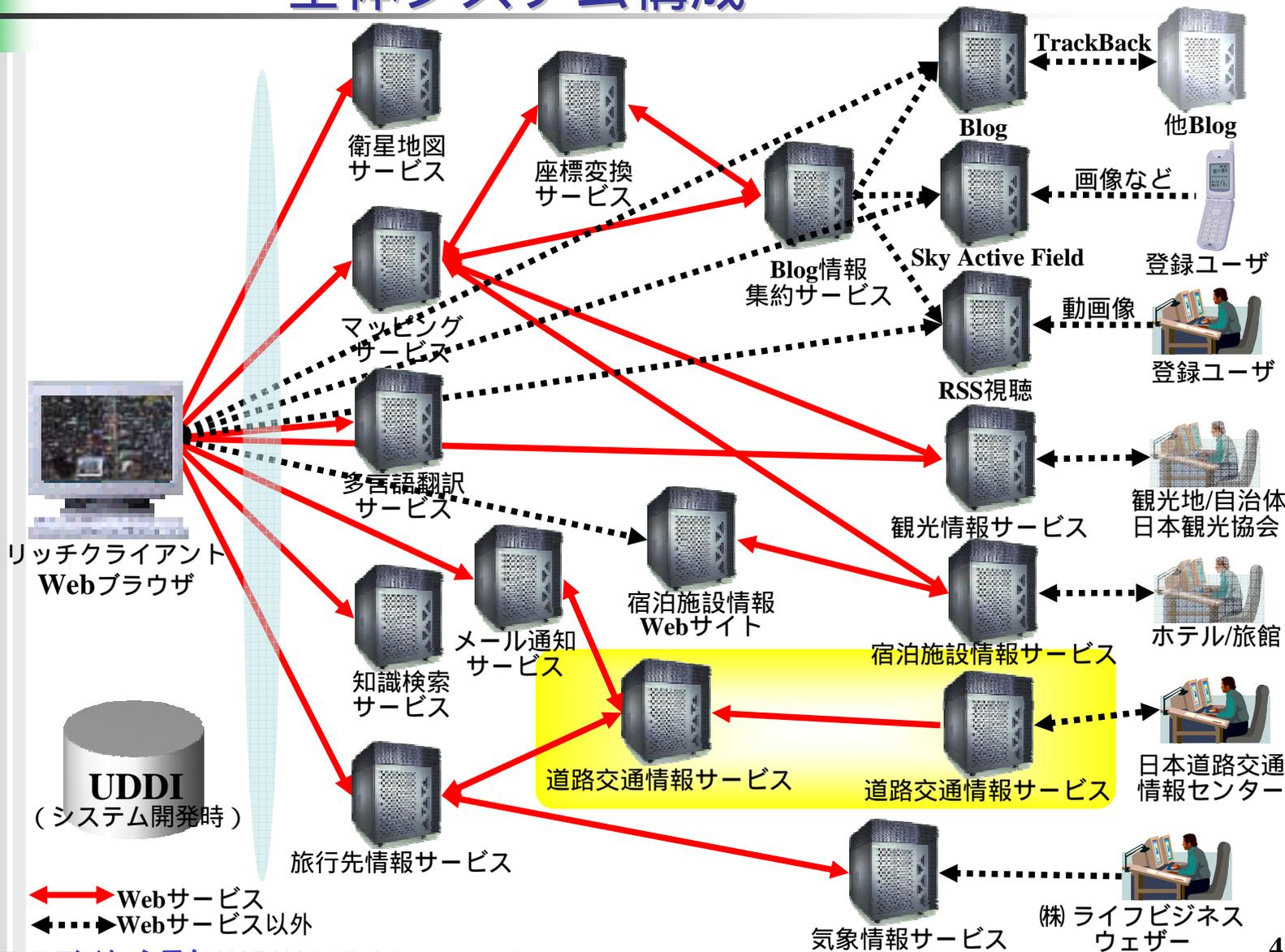
アドソル日進株式会社 荒本道隆

アジェンダ

- 道路交通情報Webサービスの目的
- 全体システム構成
- 道路交通情報Webサービス
- デモでの利用方法
- 道路交通情報の流れ
- Jシステムについて
- 道路交通情報センターとの接続について
- 道路交通情報について
- Webサービスの作成について

- Jシステムの情報を多目的に利用
 - 仮想一次事業者として、道路交通情報センターから道路交通情報XMLを受信し、色々な用途に使用するモデルを実証する
- 機能を限定する事で、インターフェイスを簡略化
 - 道路交通情報センターに検索をリクエストしようとする、XPath/XQueryでできるけど....
 - インターフェイスには、iPlatプロジェクト共通のスキーマを使用
- 応答速度をできるだけ早く
 - ユーザーインターフェイスでの利用が前提
 - データ件数が非常に多いので、検索速度が心配

愛地球博試験運用向け 全体システム構成

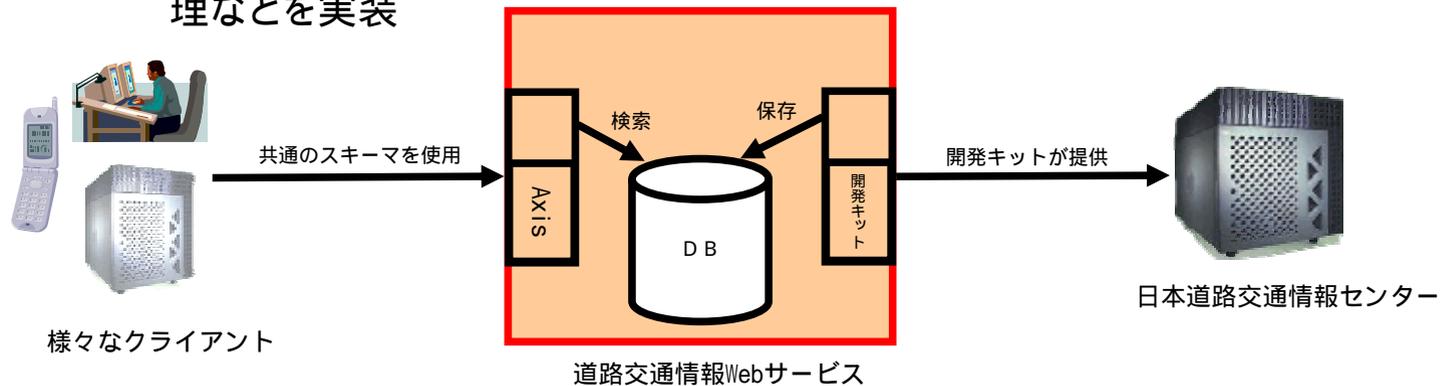


■ 動作環境

- Windows 2003
- Sun JDK1.4.2
- Apache 2.0.53 + Jakarta Tomcat 4.1.31
- Axis 1.1
- PostgreSQL 8.0.2
- 開発キット(道路交通情報センター)

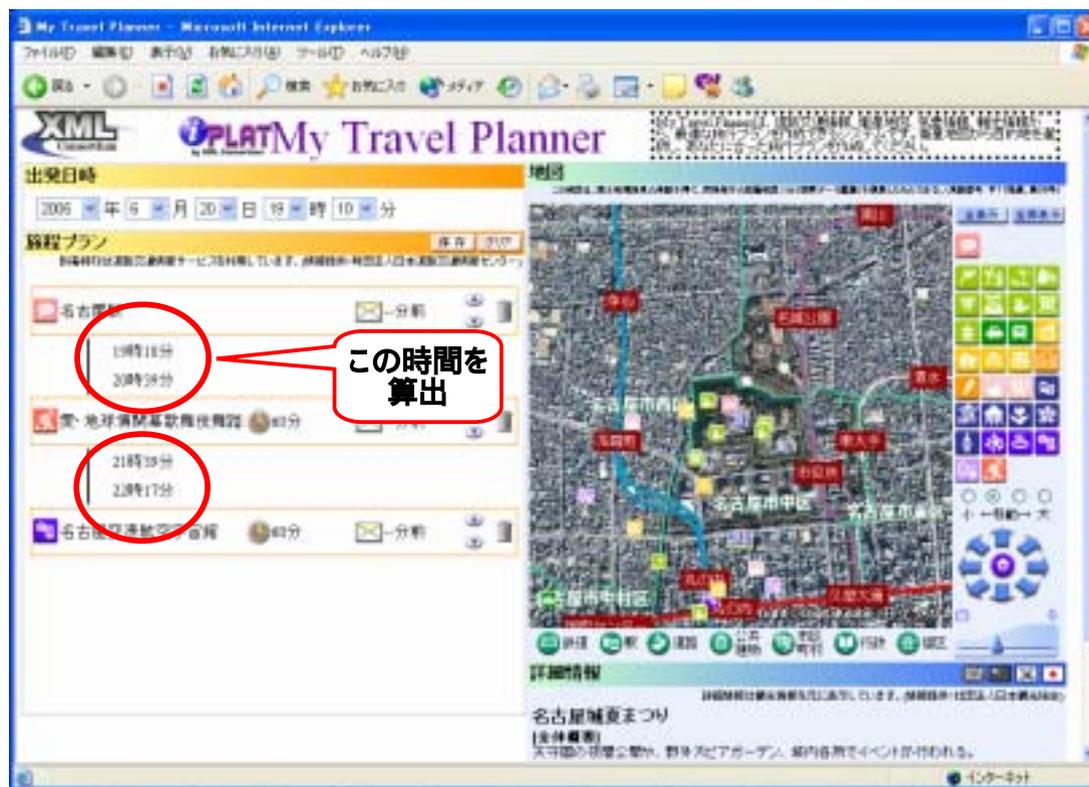
■ 開発方法

- 道路交通情報センターへのアクセス
 - 開発キットのサンプル(Java版)を流用
- 道路交通情報Webサービス
 - WSDL から Axisの WSDL2Java でスケルトンを作成し、impl に検索処理などを実装



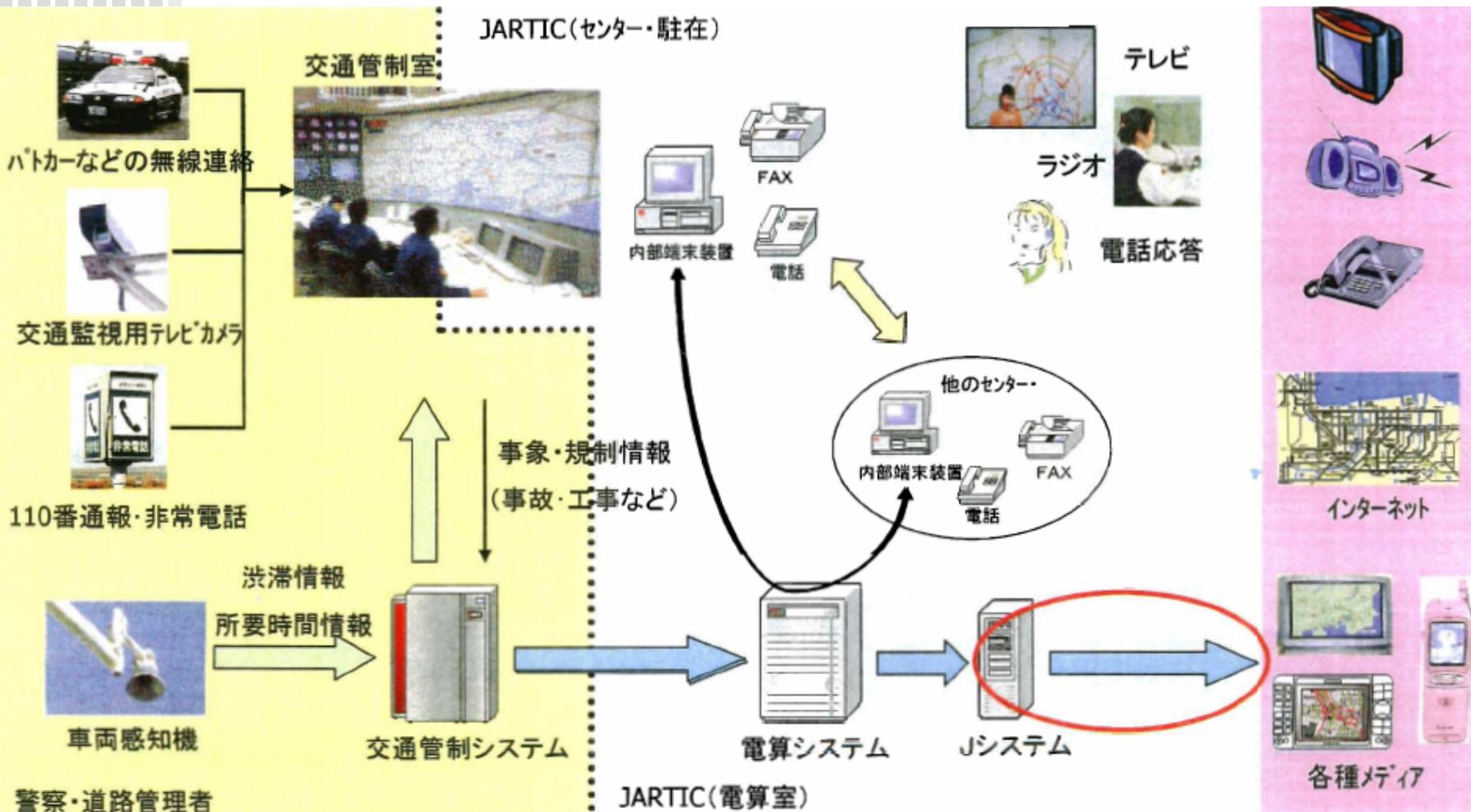
デモでの利用方法

- プランを立てる時に、移動時間の目安を算出



- メール通知直前に、最新の実測値を参照
 - メール通知サービスがメールを送る直前に補正

道路交通情報の流れ



出典：財団法人本道路交通情報センター（JARTIC）

Jシステムについて

特徴

Jシステムは、道路交通情報を利用しやすいデータ形式で、オンラインにより提供する汎用的なシステム

- ✓オンライン（専用線）で24時間提供
- ✓様々なメディアで利用可能なデータ形式の情報を提供
- ✓情報は、1分周期又は5分周期で更新され、常に最新の情報を提供

情報の種類

情報の種類	内容
渋滞・規制情報	高速道路等及び一般道路の通行止め、交通渋滞、交通事故、道路工事等による交通規制、降雨や降雪により速度規制等に関する情報
所要時間情報	高速道路等及び一般道路の所要時間情報
都市高速道路入り口閉鎖情報	都市高速道路の閉鎖されている入口に関する情報
駐車場情報(VICS符号型のみ)	一般道路の駐車場の位置や、満車・空車に関する情報
SA・PA情報(VICS符号型のみ)	高速道路等のサービスエリア・パーキングエリアの位置や満車・空車等の情報

5分周期で
配信される

情報の形式

情報の形式	情報の内容	用途
テキスト型	渋滞・規制、所要時間、都市高速入り口閉鎖	文字情報を表示するための用いられる。
フリガナ		50音順での検索や音声合成の基礎データとして利用に適する
簡易図形型	渋滞・規制	簡易図(デフォルト図)で表示するために用いられる。
VICS符号型	渋滞・規制、所要時間、都市高速入り口閉鎖、駐車場情報、SA・PA情報	デジタル道路地図で表示するための用いられる。

出典：財団法人本道路交通情報センター（JARTIC） 8

■ 道路交通情報XMLの取得方法

- 定時送信 (Push型)
 - 5分周期でデータが配信される
 - 大量のデータが配信されるので、必要な情報のみを蓄積・利用する
- 随時送信 (Pull型)
 - 必要な時に XPath/XQuery で検索
- 今回は、開発キットのサンプルをほぼそのまま使用
 - XMLがStringで渡されるので、DOM化して操作するコードを追加
 - まずは『データ種別』を見て、処理を振り分ける

道路交通情報XMLについて

■ 定時送信の受信データ例

データ種別
1 0 0 番台：リンク情報
2 0 0 番台：簡易図形情報
3 0 0 番台：渋滞規制情報
6 0 0 番台：所要時間情報
8 0 0 番台：都市高速道路入口閉鎖情報

始点の緯度・経度

終点の緯度・経度

最終更新日時

所要時間（秒）

```
<?xml version="1.0" encoding="UTF-8" ?>
<data datetime="200410020000">
  <_XML version="0.10">
    <_traffic-info seq="1">
      <_basic-info>
        <_priority="3">
          <_info-type="603">
            <_places>
              <_place seq="1">
                <_up-down-type>
                <_up-down-type>
                <_point>
                  <_lat="35.09.44">
                  <_long="136.55.86">
                  <_distance="0">
                    <_point-name>
                    <_point-name-kan>
                    <_point>
                    <_route>
                    <_route-name>
                    <_route-name-kan>
                    <_road-class>
                    <_route>
              <_place seq="2">
                <_up-down-type>
                <_up-down-type>
                <_point>
                  <_lat="35.09.31">
                  <_long="136.55.86">
                  <_distance="35">
                    <_point-name>
                    <_point-name-kan>
                    <_point>
                    <_route>
                    <_route-name>
                    <_route-name-kan>
                    <_road-class>
                    <_route>
              <_place>
            <_traffic-info>
              <_event-number>
              <_last-update>
                <_last-update>
                <_basic-info>
                <_travel-time>
                  <_origin-direction>
                  <_origin-direction-kan>
                  <_destination-direction>
                  <_destination-direction-kan>
                  <_duration>
                <_travel-time>
              <_traffic-info>
            <_traffic-info seq="2">
              <_basic-info>
                <_travel-time>
                  <_origin-direction>
                  <_origin-direction-kan>
                  <_destination-direction>
                  <_destination-direction-kan>
            </_traffic-info>
          </_traffic-info>
        </_basic-info>
      </_info-type>
    </_traffic-info>
  </_XML>
</data>
```

■ 提供するWebサービス

- Calendar getArriveDate(TravelPlanBasicType arriveDateRequest)
 - 出発日時を指定すると、到着日時を算出する
 - 到着日時を指定すると、出発日時を算出する
- int getDuration(TravelPlanBasicType durationRequest)
 - 出発日時か到着日時を指定すると、所要時間(秒)を算出する
- int getDistance(TravelPlanBasicType distanceRequest)
 - 2点間の距離(m)を算出する(緯度・経度により、直線距離で算出)

説明:

- TravelPlanBasicType は、iPlat用に定義した共通のスキーマ
- 指定した日時の、もっと近い過去のデータを使用する
- 未来の日時を指定した場合には、同じ曜日・同じ時間の平均値を使用する
- 日時にnullを指定した場合には、最新のデータを使用する

- 大量のデータを格納
 - 1日のデータ件数(愛知県内のみ)
 - $459 \text{件} \times 12 \text{回} / \text{時間} \times 24 \text{時間} / \text{日} = 132,192 \text{件} / \text{日}$
 - 毎回ほとんど変わらない部分をマスター化
 - 始点・終点の座標・名称、道路の名称などは滅多に変わらない
 - 1件あたりのデータ量:最大2900Byte 70Byte
 - XMLのままだと、1件あたり5KByte程度
- 検索時のパフォーマンス
 - 曜日・時間ごとの平均値を予め算出
 - 「指定された時間に最も近いデータ」を探すのは、意外に大変
 - 『select max>LastUpdate) from TravellInfo where ID=? and LastUpdate <= ?』は蓄積件数が多くなると遅くなる
 - 経路検索で頻繁に使用するマスター情報は、すべてメモリ上に読み込んでおく

— 開発を終えて —

- 他システムとのインターフェースの部分は、まったく苦勞しなかった
 - 開発キットのサンプル、WSDLが提供されたおかげ
- 道路交通情報XMLのスキーマが提供されていない
 - XMLをDOM化して自前で操作しなければならない
- とにかく、検索速度の高速化が大変だった
 - 最初は早かったのに、データの量が増えてくると遅くなっていく
 - DBのバッファ、別プロセスからのDBへの書き込みなど、DBアクセスの検索速度が一定にならない
 - たまたまキャッシュが効いていたために、「早い」と勘違いした事もあった
- 経路検索を独自に開発してみたものの...
 - 『所要時間情報』がある経路から探すとなると候補が少ない
 - 近い場所や、大きな道路がない場所は、該当する候補がない
 - 今回は、候補がない場合には、2点間の距離を算出し、想定速度で割ることで時間を出した