Service-Oriented Architecture for Automotive Cloud Service Systems

Akihito Iwai
Software Tech. R&D Dept.
Corporate ePF Division
DENSO CORPORATION

DENSO

Outline

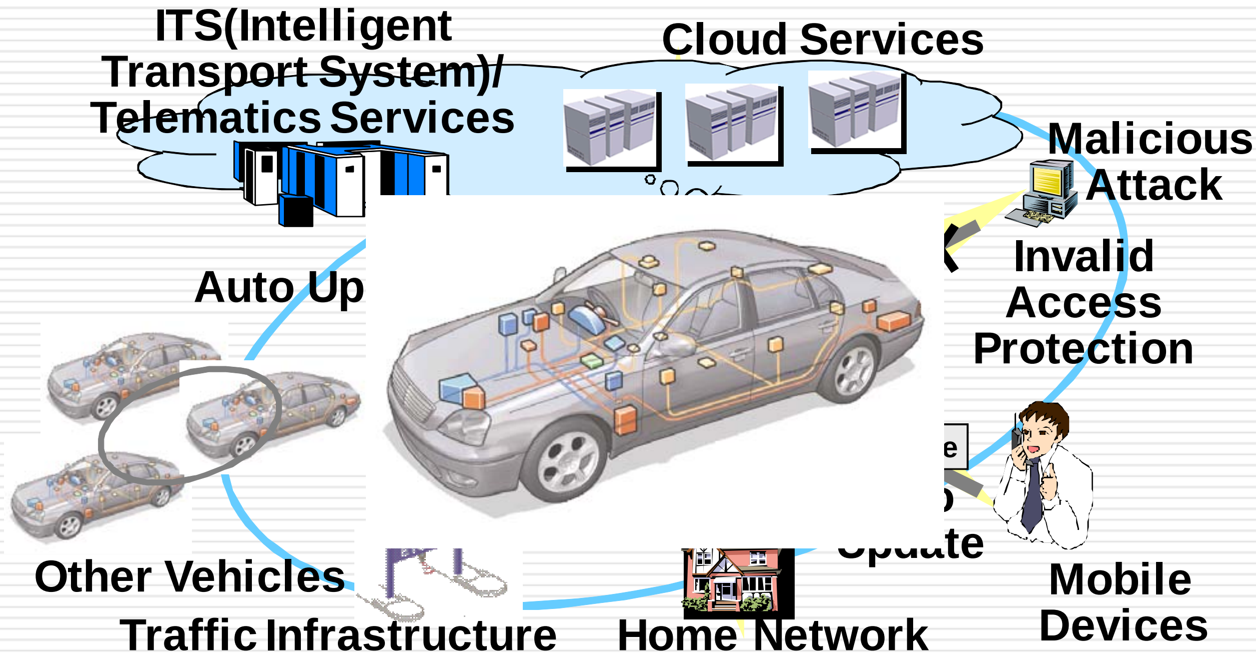
- Background: Automotive Cloud Service Systems (*ACSS*)
- Key Challenges for ACSS
- Related Works
- ACSS Architecture
- Case Study
- Discussions
- Conclusions and Future Work

Background: Automotive Cloud Service Systems (*ACSS*)

- Automotive Software Systems become more large-scale and complex year by year
- ⇒ Evolving to "***Automotive Cloud Service Systems: ACSS***"
  - Vehicle + "***Traffic Infrastructure***"
  - + "***IT service***"
  - + "***Power Grid***"
  - ………

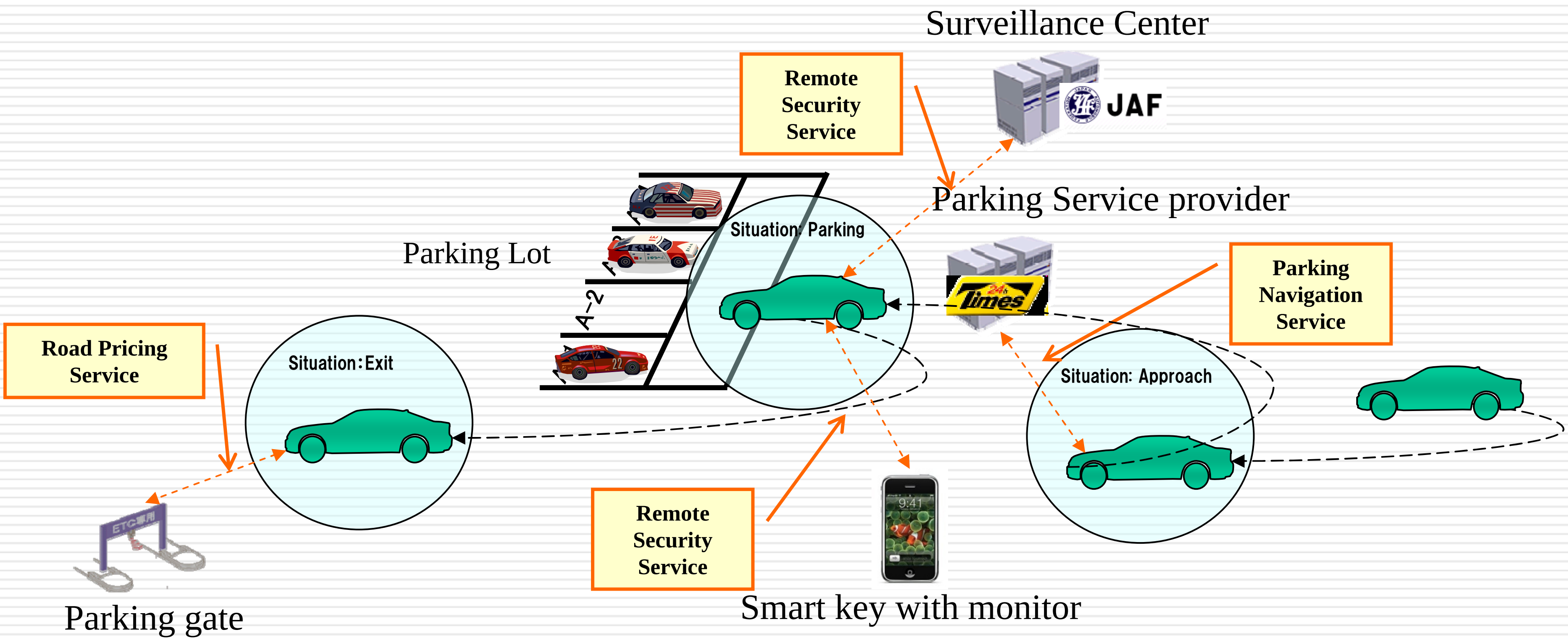
ITS, Telematics

Smart Grid



Example of Service: Intelligent Parking Service

- Car, service provider and mobile phone work collaboratively to provide parking navigation, remote security and road pricing.
- Car provides appropriate services according to the requirements of the situation autonomously.

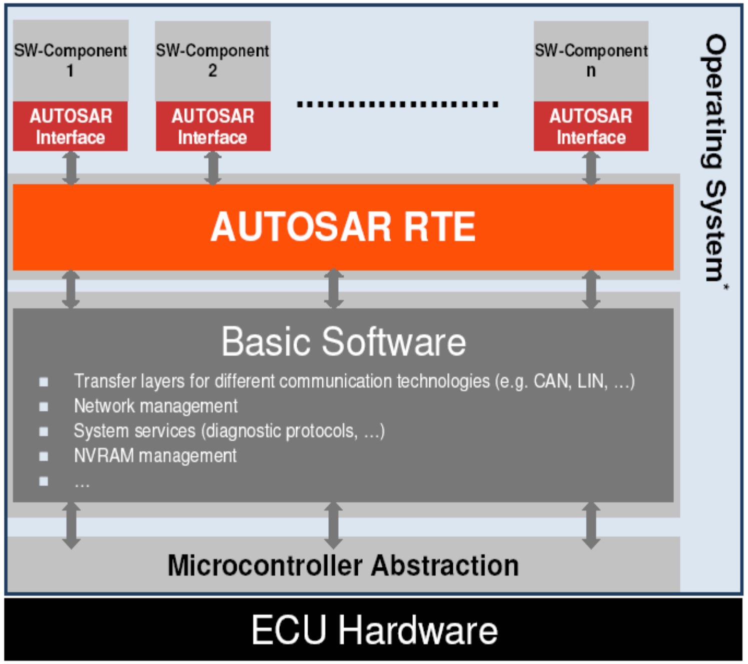


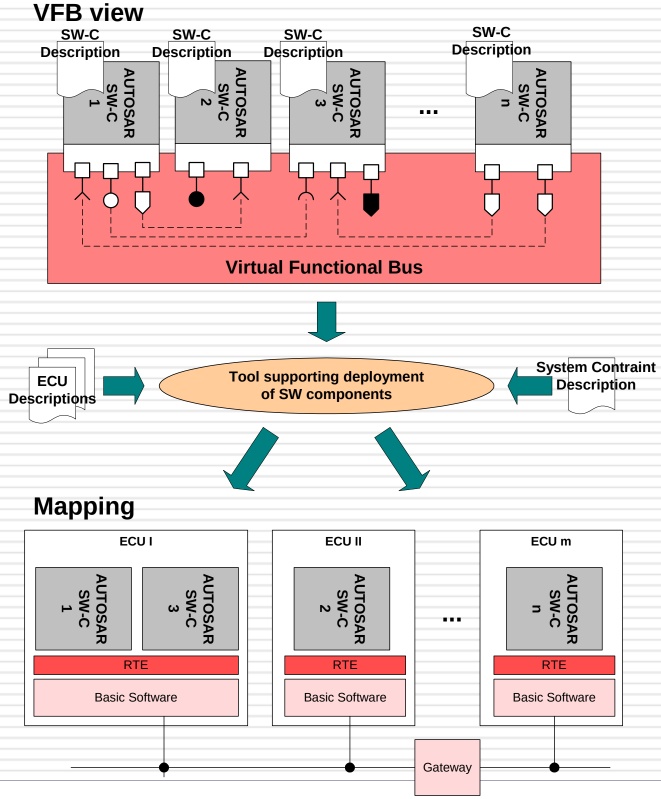
Key Challenges for ACSS

- **Guarantee of Real-time Performance**
  - The automotive software system is a real-time embedded system which is imposed strict time constrains
    - signal processing of Sensor/Actuator within the determined time interval
- **Assurance of Safety and Reliability**
  - The automotive software system is a safety-critical system which can be fatal to human life
    - Electronic Control Unit (ECU) needs high safety and reliability
- **Embracing the Short Product Life Cycle**
  - The product cycle of the computing devices in-and-out of vehicle is much shorter than that of the automotive devices
    - Automotive service-related standardizations have not matured to practical use yet
      - NGTP and GENIVE for telematics systems

Related Works

- **SOA: Service-Oriented Architecture**
  - Architecture to integrate the information services and communication services seamlessly
  - Dealing with all information as ***"Everything is a Service"***
- **AUTOSAR (Automotive Open System Architecture)**
  - Open and standardized automotive software architecture
  - **Component based** software design model



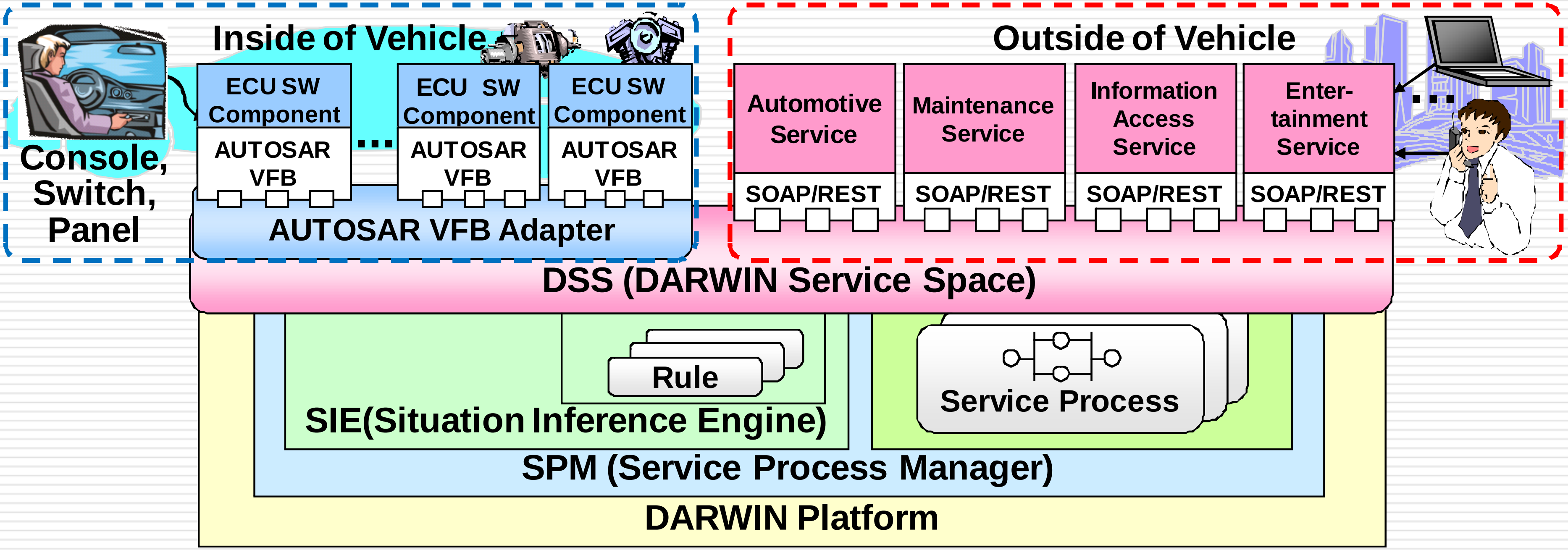


- **Service Orchestration in-and-out of Vehicle**
  - To collaborate between ECU Software Components and external services in safe and within time constraints
  - To optimize service processes for a variety of user requirements and the vehicle context
  - To continue to run even under unstable network connection
    - momentarily communication or power blackout
- **Information Sharing over Networks**
  - To provide service interfaces for sharing information
  - To remove and modify information with safe after purchased
- **Non-Functional Requirements**
  - To keep fault tolerant against unpredictable failures
  - Low cost for in-vehicle terminal

DENSO

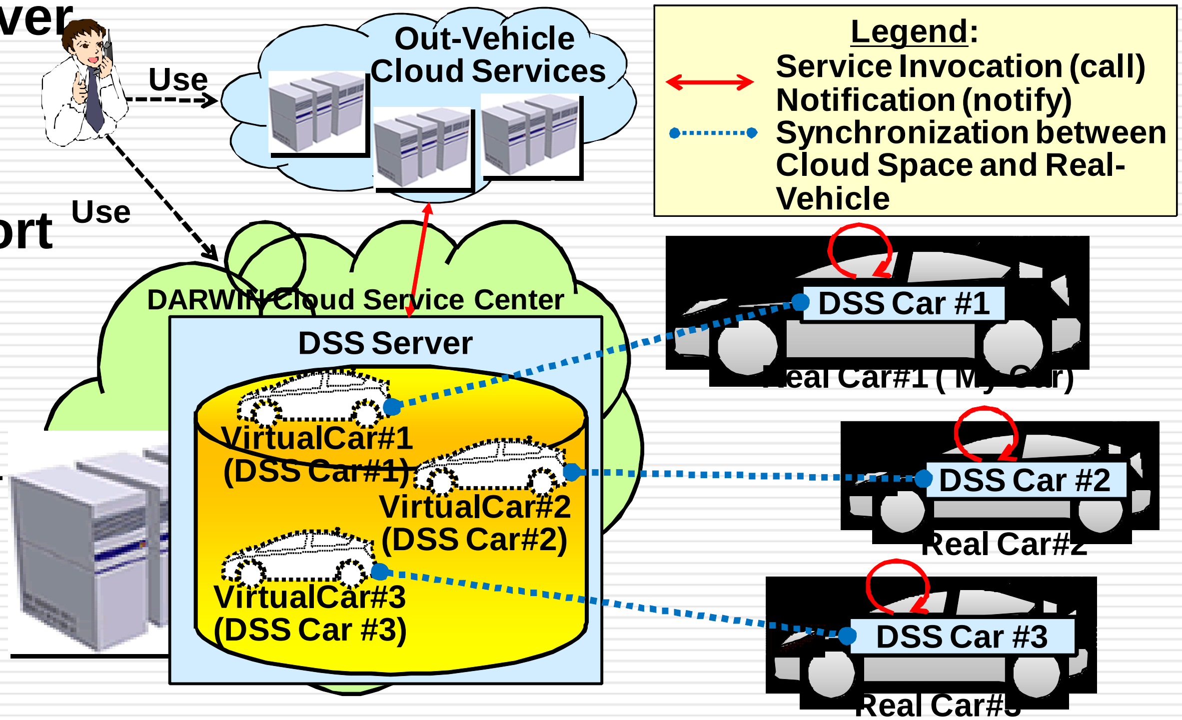
ACSS Architecture: What is DARWIN Architecture?

- **DARWIN platform contains two main software modules**
  - DSS (DAWIN Service Space)
  - SPM (Service Process Manager)
- **Provide service/component Integration/Interoperability**
  - ECU Software Components (ECU SWC)
  - Services
  - AUTOSAR VFB Adapter included in DSS
  - Service Link Protocol (see the details later)



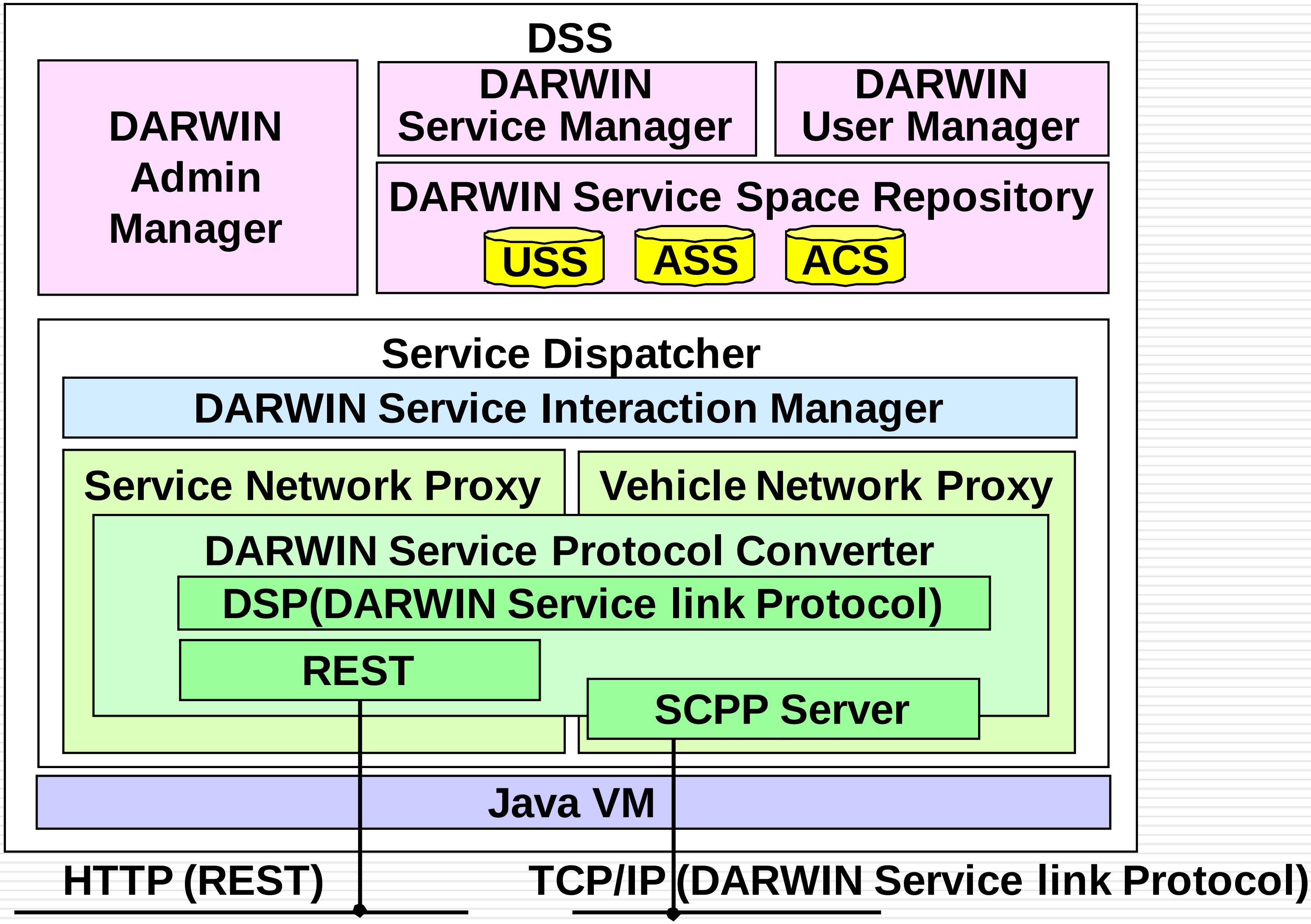
ACSS Architecture: DSS(DARWIN Service Space)

- **Challenges**
  - Meet the requirements on the high safety, reliability and real-time performance for automotive applications
    - Exiting technologies such as Java Space, CORBA are not met
- **Key features**
  - **Cloud based DSS server**
    - Encapsulation of interaction among Real Cars
  - **AUTOSAR VFB support**
    - ECU Software components communication with other software components or in-out-vehicle services
  - **QoS management mechanism**
    - Prioritized messing service



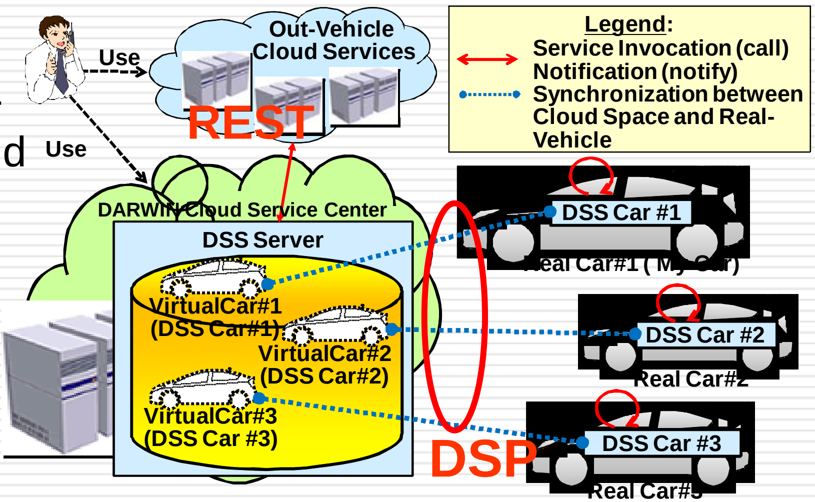
DENSO

Detailed Software Architecture of DSS in Vehicle



DENSO

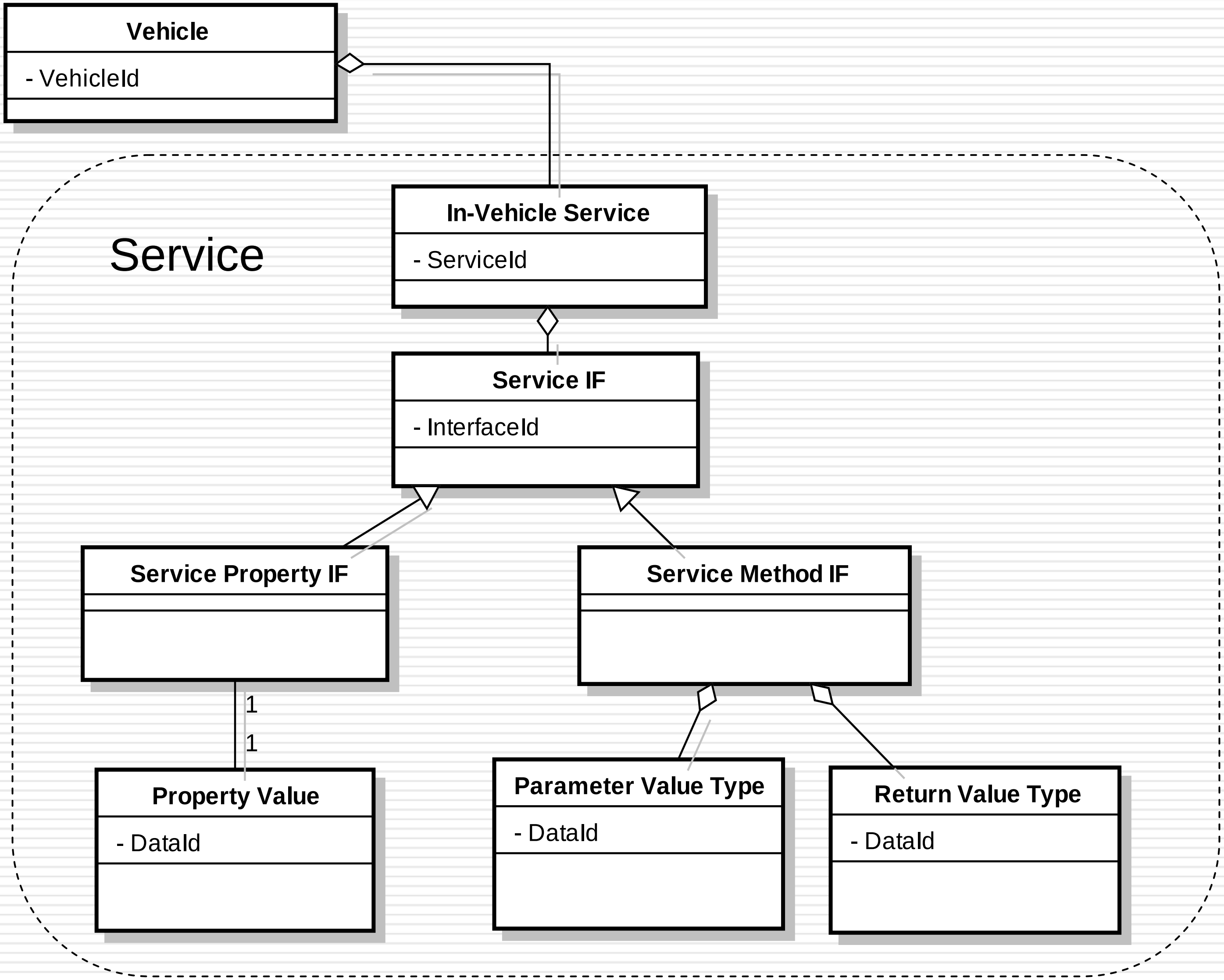
- **Two kinds of service link protocols are supported**
  - For communication between Datacenter and in-vehicles
    - **DSP（*Darwin Service Link Protocol*）**
      - Original Protocol
      - Compatible with subset of NGTP（New Generation Telematics Protocol）
      - High performance and lightweight protocol by data compression
  - For communication between Datacenter and out-vehicle services
    - **REST (*REpresentational State Transfer*)**
      - Lightweight vs. SOAP
      - HTTP base, resource identified by URL
      - Simple basic API: GET, PUT, POST and DELETE
      - Extension for management of in-out-vehicle service base on REST
        - » Car ID, Service ID, Interface ID



URL description example:

http://darwin.jp/api/001/vehicle/{*car ID*}/services/{*service ID*}/interfaces/{*interface ID*}/call

REST API structure



DENSO

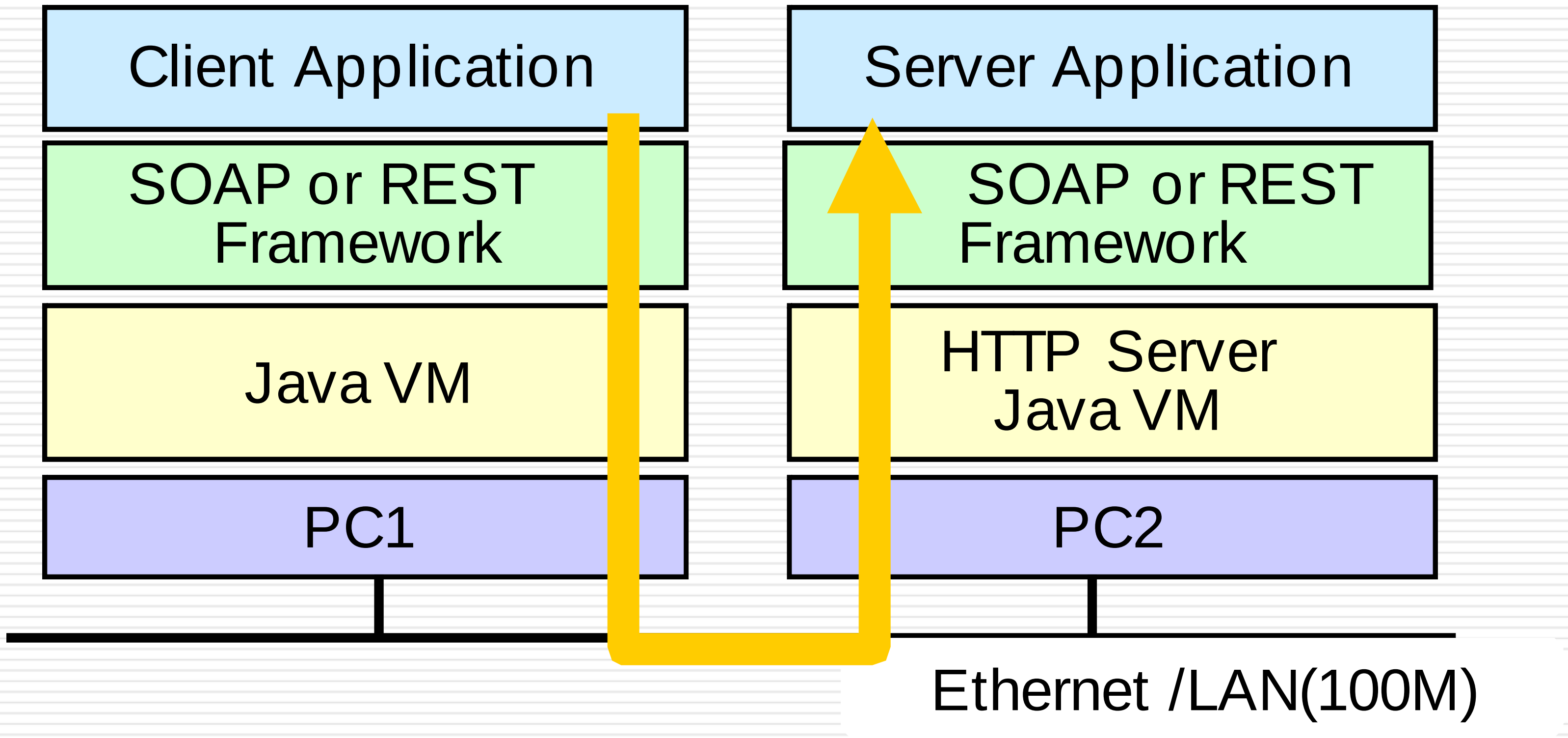
Example of DARWIN API

ServiceName	Service Id	API	descrition
Device Application Service	AppDevice	setMainLight	Control front main light.
		setFogLight	Control front fog light.
		getMainLightStatus	Collect current main light status.
Auto driving Service	AppAuto	setAutoDriveMode	Set automatic driving mode. (e.g. Auto-Parking)
Probe Information Service	AppProb	getSpeed	Collect current speed.
		getBattery	Collect remain battery capacity.
		getBatteryOutput	Collect battery output voltage.
	AppSon	getClearanceSensors	Collect vehicle clearance conditions.

See DARWIN API Specification in details

DENSO

Performance Evaluation



Configuration of Evaluation Environment

Specification on performance evaluation

		PC1 (Client)	PC2 (Server)
CPU		Intel Pentium M 1.70GHz	Intel CoreDuo 1.83GHz
Memory		1.0 GB	1.5 GB
OS		Linux: Ubuntu 9.0.4 (2.6.28-generic)	Linux: Ubuntu 9.0.4 (2.6.28-generic)
Middle ware	SOAP	Axis 1.4.1	Axis 1.4.1
	REST	Restlet2.5	Restlet2.5
	HTTP	-	Apache Tomcat 6.0.2
Java VM		Sun Java1.6.0_14	Sun Java1.6.0_14

DENSO

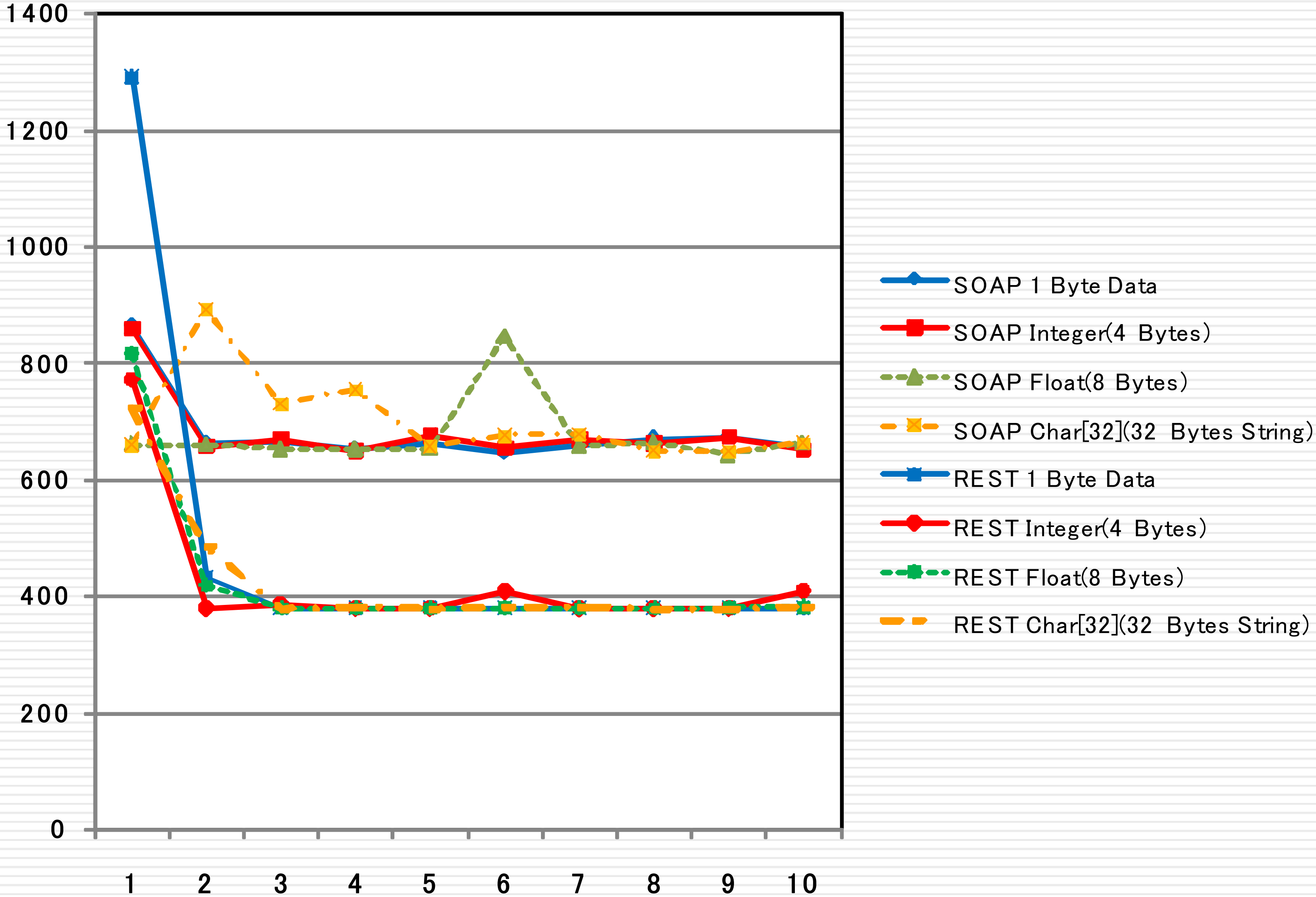
API for Service Call

*Note: Bytes

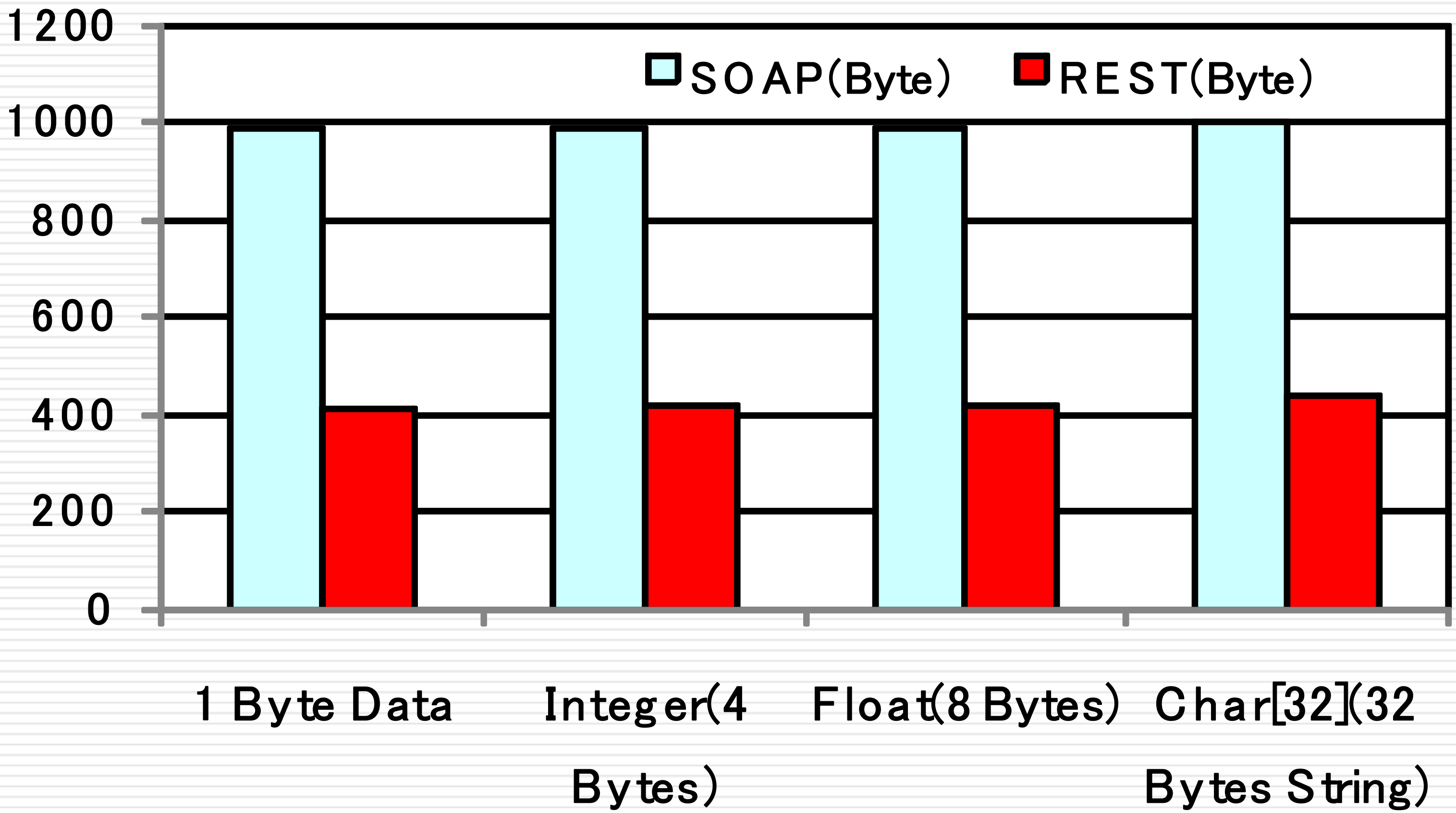
#	Field	Size*	Description
1	MESSAGE_VERSION	4	Number of message version
2	MESSAGE_ID	15	Message ID in the system including DSS server and vehicles, numbed cyclically
3	MESSAGE_TYPE	7	Structure type of protocol data
5	SERVICE_CTRL_SIZE	4	Data size for service control field, always ‘0’ if not necessary
6	BODY_FIELD_SIZE	4	Data size for body field, saved by compressed transmit size, always ‘0’ if not necessary
7	TIME	23	Start time of service call (specified by caller), usage for validate check in caller side, time strings with “yyyy-MM-dd HH:mm:ss:SSS” format
8	SOURCE_USER_ID	7	User ID for service consumer, NULL if not user
9	SOURCE_CAR_ID	7	Car ID for service consumer, NULL if not vehicle
10	TARGET_SERVICE_ID	7	ID for identifying service of service provider, NULL if control command of DSS server
11	TARGET_SERVICE_INTERFACE_ID	7	ID for identifying service interface of service provider, NULL if control command of DSS server
12	DATA_FIELD	Variable	Transmit data value (data size depends on request parameters of caller)

DENSO

Message Processing Time for Service Call



Payload of Communication Data for Service Call

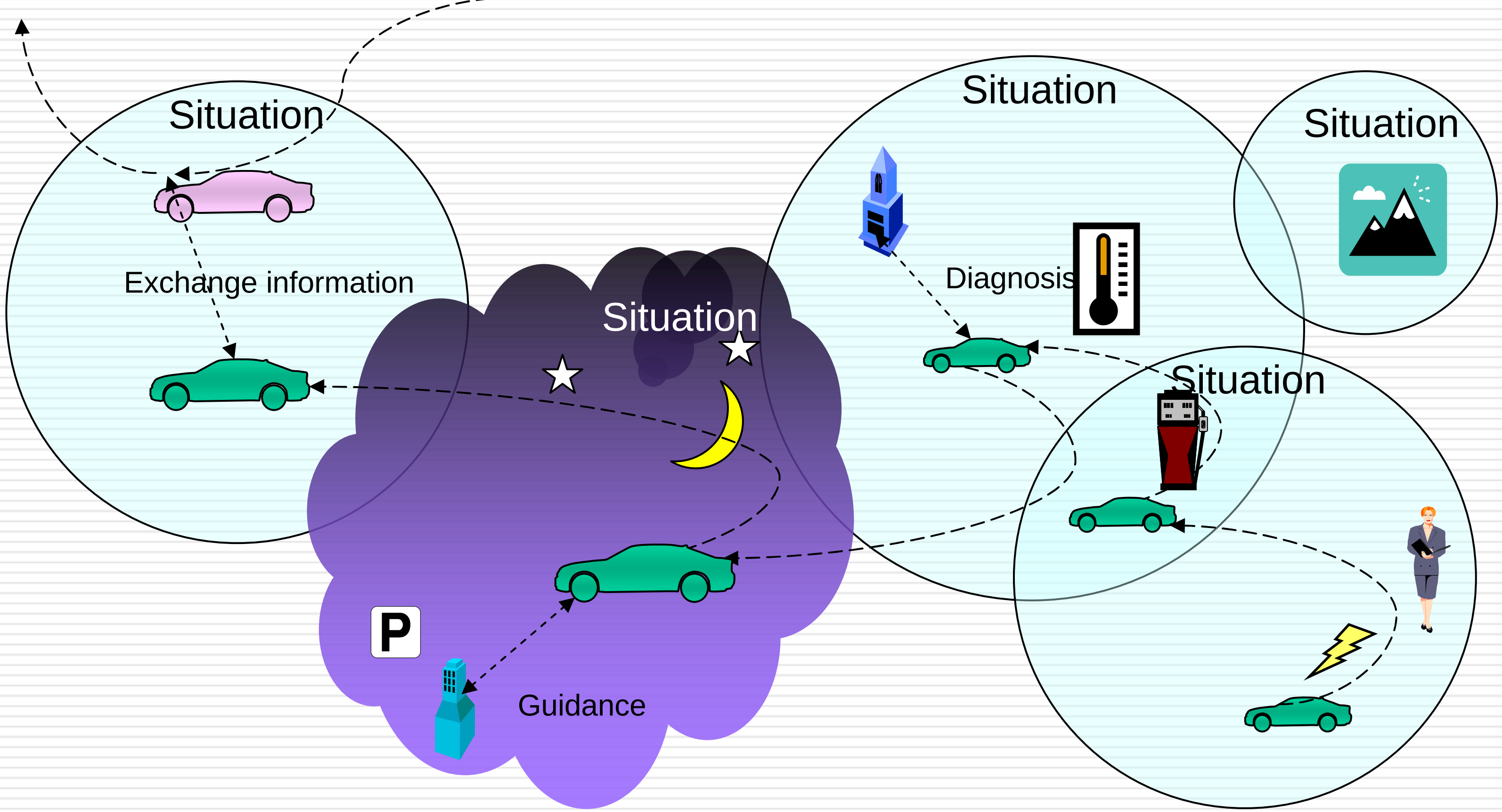


DENSO

ACSS Architecture: Darwin Service Concept

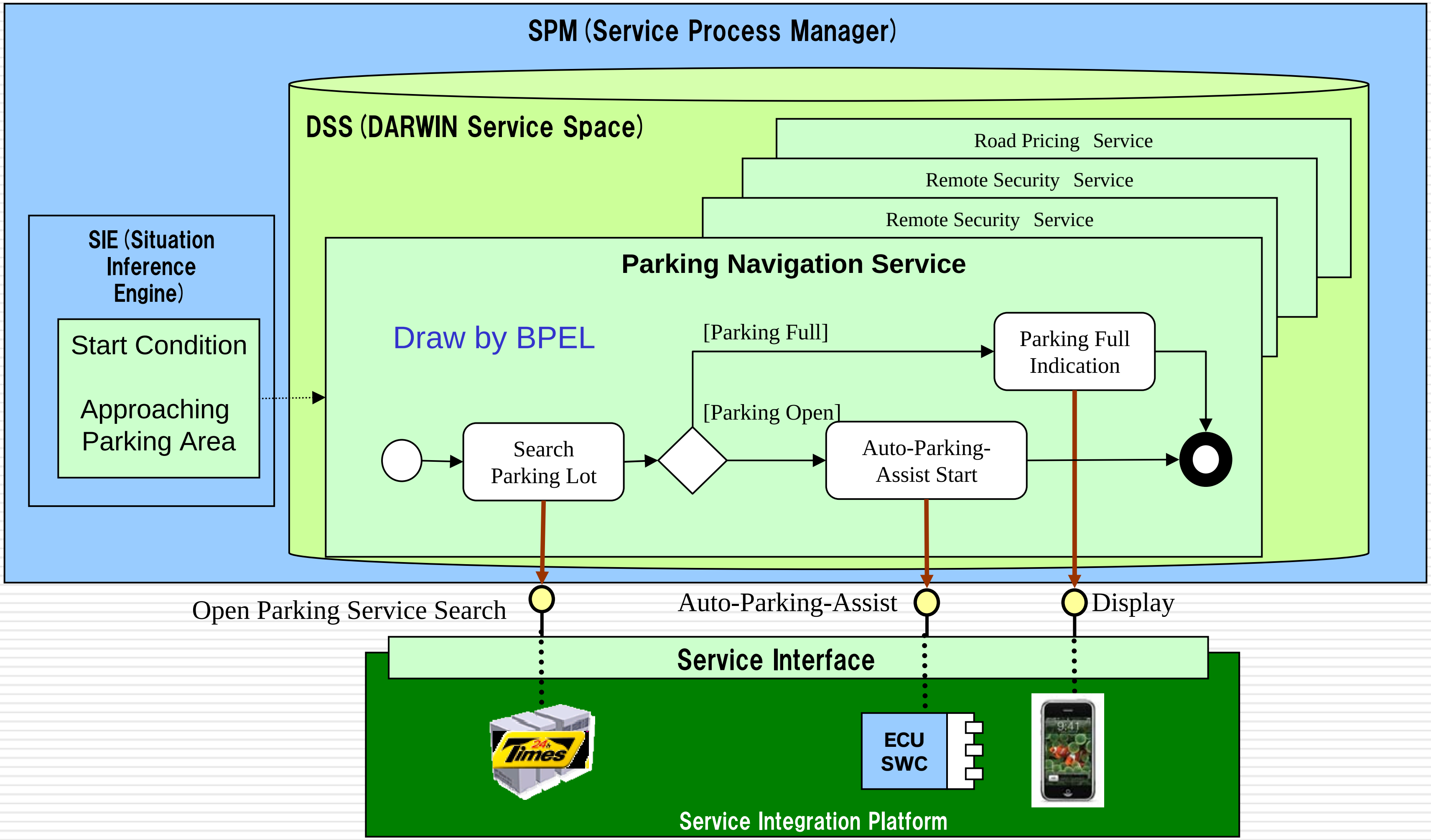
Situation = Space（Where） & Time（When）

- Situation Matching
  - Car moves through various situations.
  - Service Integration platform executes appropriate services according to the requirements of the situation autonomously.

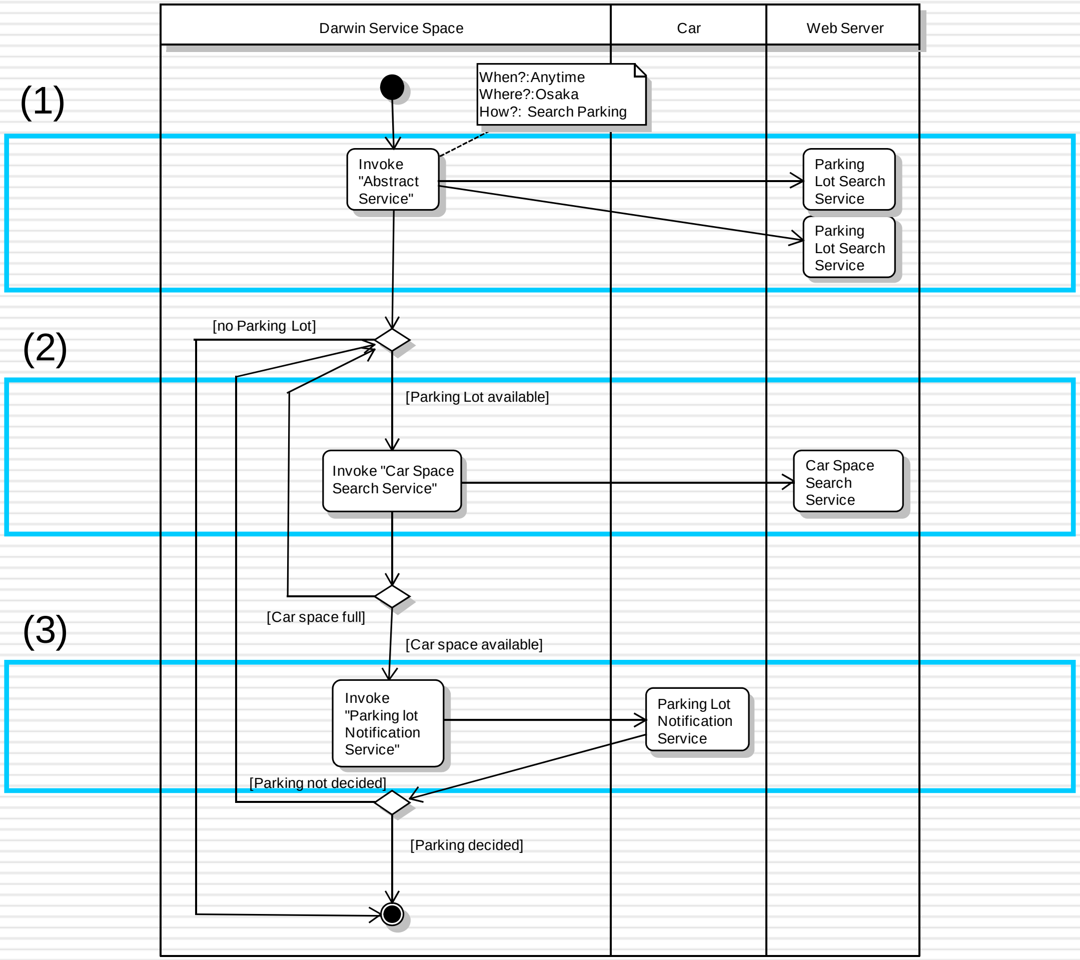


ACSS Architecture: SPM (Service Process Manager)

- Easy Integration of services using Service Interface



Attempt to minimally extend BPEL



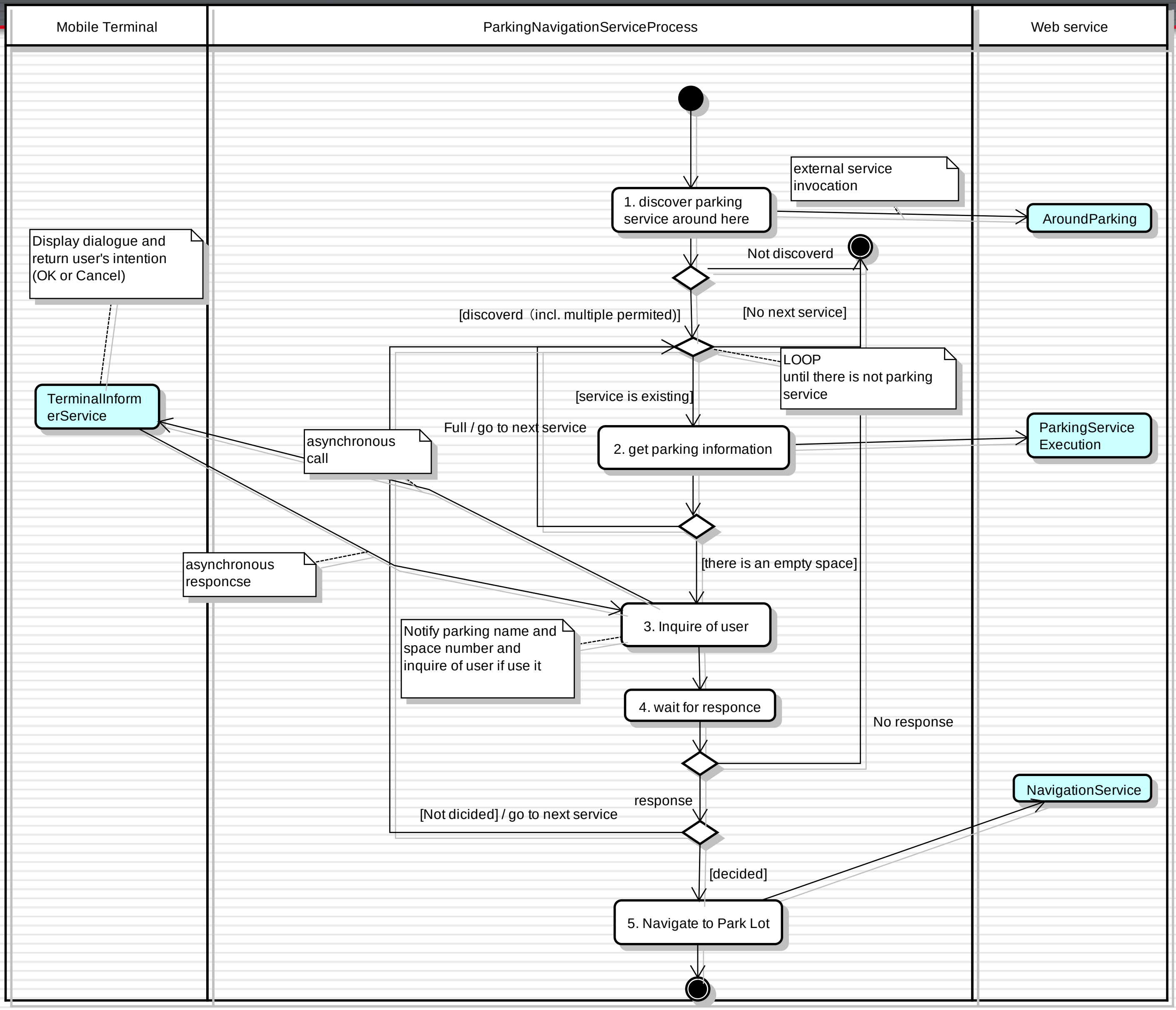
Example of BPEL description

(1)
```
<invokeAbstractService when="always" where="area:Osaka" what="Search parking" execute="all" timing="start">
  <params>
    <param type="int">latitude</param>
    <param type="int">longitude</param>
  </params>
  <return type="string">ParkingServiceName</return>
</invokeAbstractService>
```

(2)
```
<invoke name="InvokeNotifyEmptySpaceNumber" partnerLink="ParkingServer" operation="GetEmptySpaceNumber" portType="GetEmptySpaceNumberPT" inputVariable="ParkingServiceName" outputVariable="ParkingNumber">
</invoke>
```

(3)
```
<invoke name="InvokeCheckParkingCar" partnerLink="CAR" operation="CheckParkingCar" portType="CheckParkingCarPT" inputVariable="ParkingNumber" outputVariable="bParkCar">
  <toParts>
    <toPart part="partnerLinkName" toVariable="ParkingServiceName" />
    <toPart part="partnerLinkName" toVariable="ParkingNumber" />
  </toParts>
</invoke>
```

DENSO


Mobile Terminal
ParkingNavigationServiceProcess
Web service
external service
invocation
1. discover parking
service around here
AroundParking
Display dialogue and
return user's intention
(OK or Cancel)
Not discovered
[discoverd（incl. multiple permited)]
[No next service]
LOOP
until there is not parking
service
TerminalInform
erService
[service is existing]
Full / go to next service
asynchronous
call
2. get parking information
ParkingService
Execution
asynchronous
responcse
[there is an empty space]
3. Inquire of user
Notify parking name and
space number and
inquire of user if use it
4. wait for responce
No response
NavigationService
response
[Not dicided] / go to next service
[decided]
5. Navigate to Park Lot

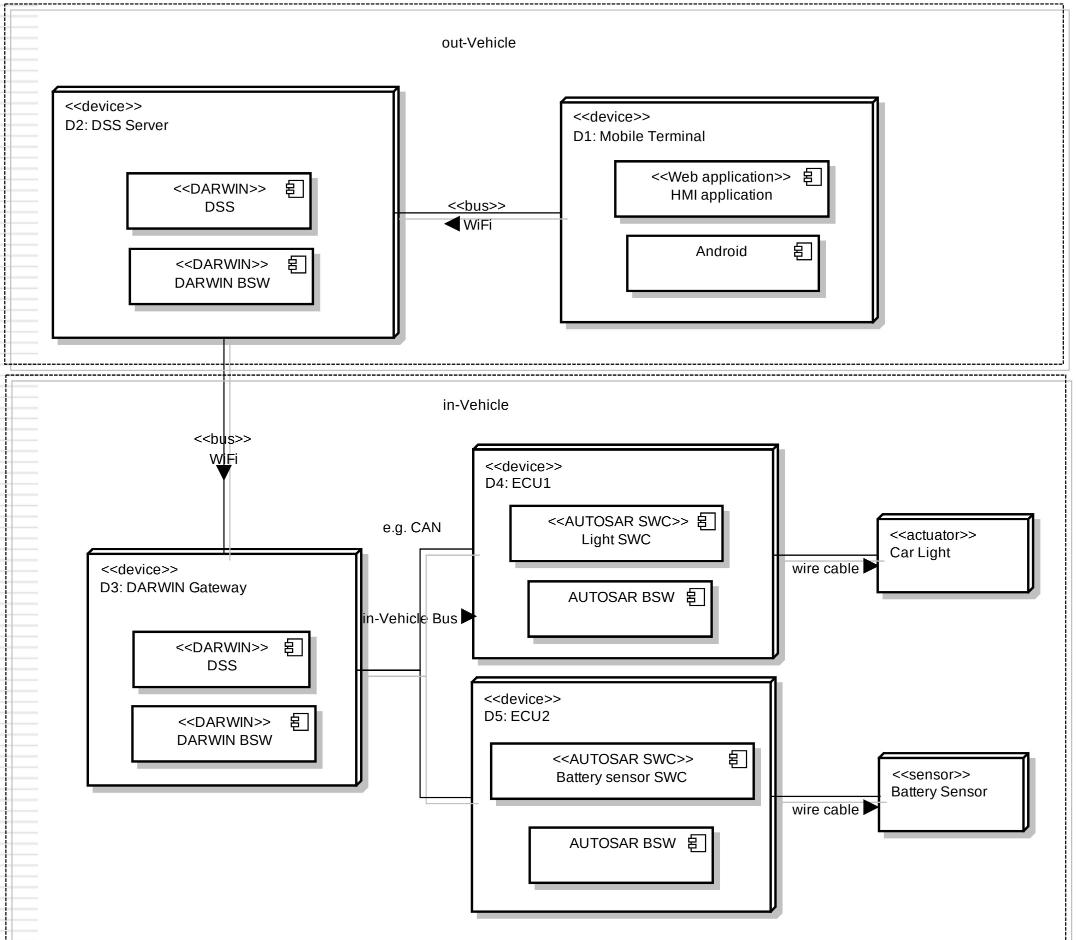
DENSO

```
<?xml version="1.0" encoding="UTF-8"?>
<process

/* Define name space, import file, desired service and etc */
 name="IntelligentParkingService"
 targetNamespace="http://enterprise.netbeans.org/bpel/IntelligentParkingService/
 DefaultServiceName"
 xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
                        :
 <import namespace="http://j2ee.netbeans.org/wsdl/TerminalInformerService"
        location="TerminalInformerService.wsdl"
        importType="http://schemas.xmlsoap.org/wsdl/"/>
                        :

/* Specify EventHandler */
<eventHandlers>
    <onEvent partnerLink="IPS_PL"
             operation="EventOperation"
             portType="ns4:IntelligentParkingServicePortType"
             variable="Event"
             messageType="ns4:IntelligentParkingServiceOperationNotify">
                        :
            <sequence name="Sequence9">
                <assign name="AssignTerminateMsg">
                        :
                </assign>
                <reply name="ForceTerminateReply"
                       partnerLink="IPS_PL"
                       operation="IntelligentParkingServiceOperation"
                       portType="ns4:IntelligentParkingServicePortType"
                       variable="IntelligentParkingServiceOperationOutTerminate"/>
            </sequence>
        </scope>
    </onEvent>
                        :
</eventHandlers>

/* Specify Sequence */
<sequence>
    <receive name="Start"
             createInstance="yes"
             partnerLink="IPS_PL"
             operation="IntelligentParkingServiceOperation"
             xmlns:tns="http://j2ee.netbeans.org/wsdl/IntelligentParkingService"
             portType="tns:IntelligentParkingServicePortType"
             variable="IntelligentParkingServiceOperationIn">
                                                        :
    </receive>
                        :
</sequence>
```

Case Study: Evaluation of DARWIN with a Prototype

We evaluated our concept of DARWIN architecture using prototype system with use cases

Configuration of Prototype System



Two usage scenarios

***Scenario 1*: Car light remote control**

Service Type: Calling in-vehicle service from out-vehicle service

Service Description: A user remotely controls car light on/off using the mobile terminal (Android Phone).

***Scenario 2*: Battery charging monitor**

Service type: Providing the vehicle information from in-vehicle to out-vehicle

Service Description: A user remotely monitors Battery sensor information in-vehicle using the mobile terminal (Android Phone).

DENSO

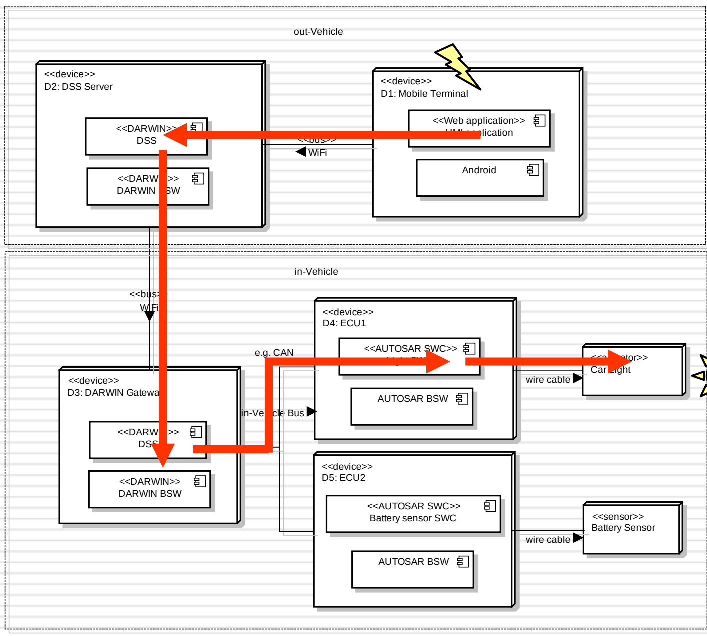
Specification of Prototype System

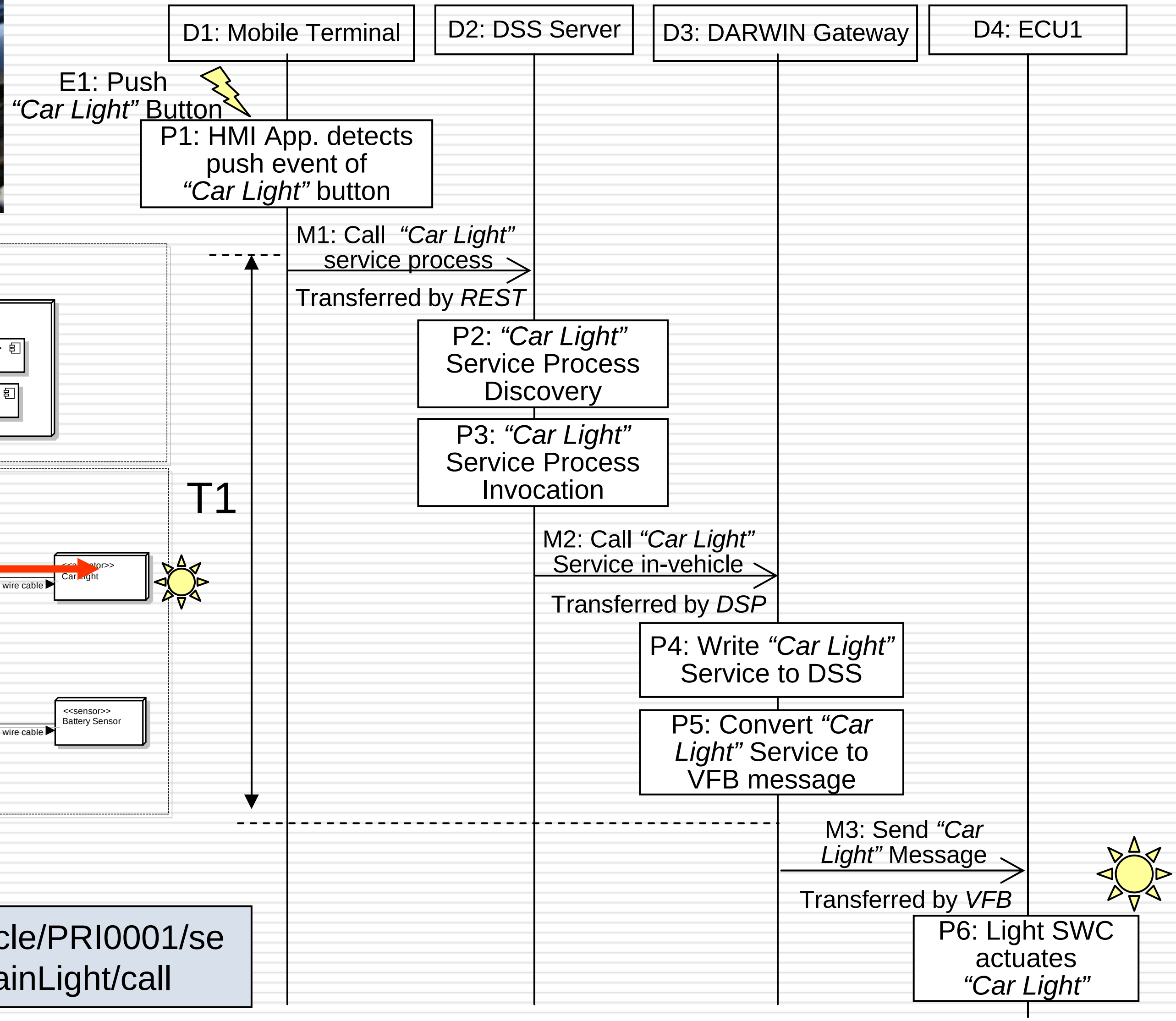
Table 1 Specification of Prototype System

Device	CPU	Memory	Software
D1: Mobile Terminal	QUAL COMM MSM7201a, 528MHz	Flash 512MB, SRAM 192MB	Android R1.6
D2: DSS Server	Intel Core DUO2 3GHz	2.5G	Debian5.0 Sun Java Ver. 1.6.0_14 Apache Tomcat Ver. 6.0.20, SQLite
D3: DARWIN Gateway	VIA C7 1GHz	1GB	Debian4.0 Linter
D4: ECU1	CUSTOM CPU	512KB	AUTOSAR Basic Software Release 3.0
D5: ECU2	CUSTOM CPU	512KB	AUTOSAR Basic Software Release 3.0

DENSO

Case Study: *Scenario 1: Car light remote control*



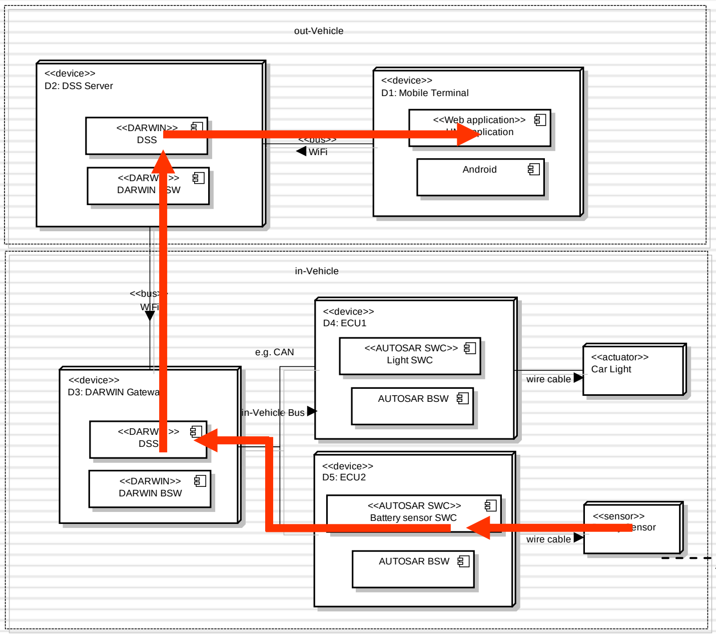


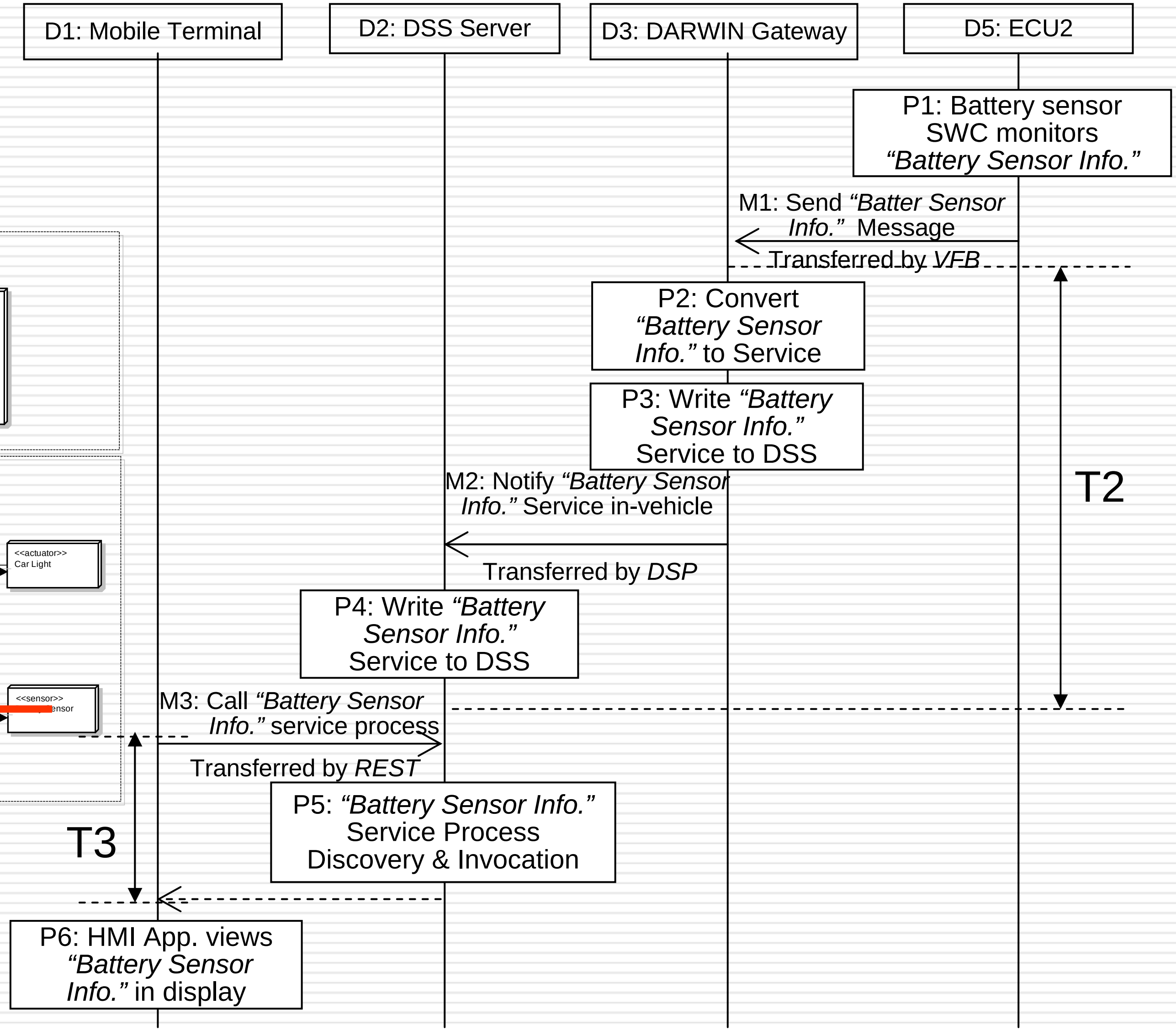
"Car Light" API:

http://darwin.jp/api/001/vehicle/PRI0001/services/SwcLoc/interfaces/MainLight/call

DENSO

Case Study: *Scenario 2: Battery charging monitor*





DENSO

Performance of service call in prototype system

Table 4 End-to-End Response Time of Service Call

Scenario	Min.	Max.	Ave.	STD
T1	139	544	162.8[205.1]	15.9[120.7]
T2	91	524	192.9[212.4]	131.1[135.4]
T3	105	433	259.3[266.6]	93.2[90.3]

Note: [] indicates the statistics including the first attempt.

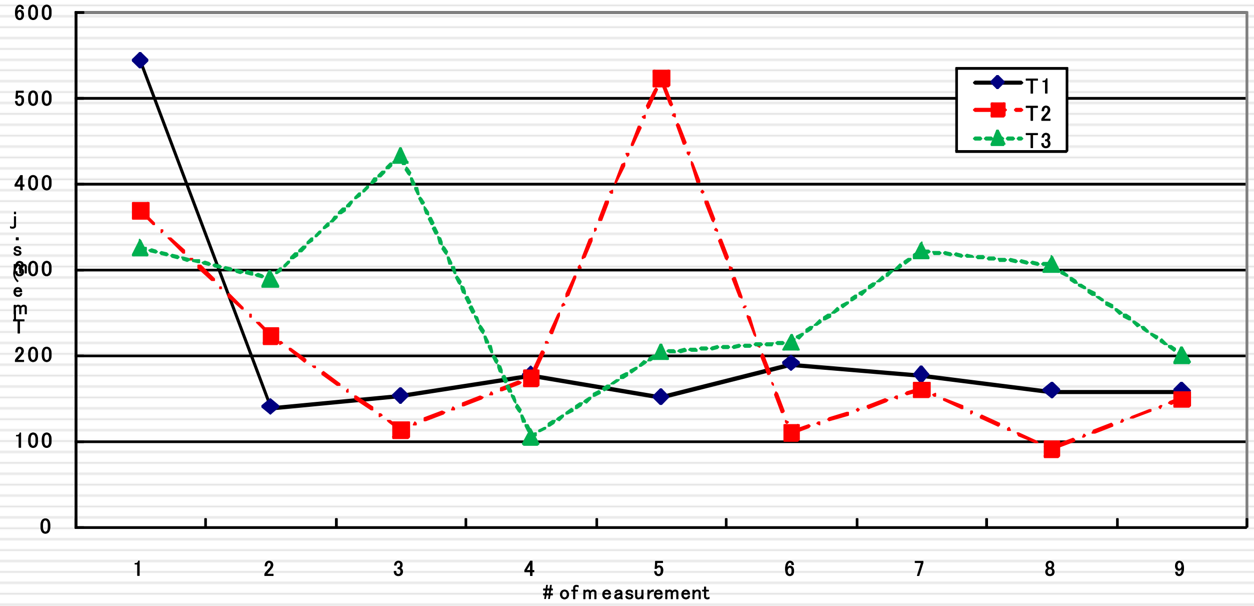


Fig. 15 Propagation Time in Prototype System

DENSO

- **Two advantages of DARWIN architecture**

1. **Seamless integration between in-vehicle and out-vehicle services**
   - DSS supports integration by standard service messaging protocols such as SOAP and REST
   - SPM provides a mechanism how to integrate and manage different service processes seamlessly

2. **Evolutional collaboration with conventional architecture such as AUTOSAR component architecture**
   - DSS works as a service bus like ESB (Enterprise Service Bus), enabling to coordinate protocols and interfaces
   - To collaborate with existing software components through AUTOSAR VFB

Demo: Emulator Tool

- **Conclusions**
  - **Challenges of ACSS**
    - Challenges of our concept of ACSS (Automotive Cloud Service System) is discussed
  - **DARWIN Architecture and its Proof of Concept**
    - We propose SOA(Service-Oriented Architecture) for ACSS named DARWIN
    - DSS(DARWIN Service Space) makes it easy for service provider to develop new automotive cloud services
    - Validated the DARWIN architecture with a prototype and two case studies
  - **Contribution of our ACSS Concept**
    - New automotive software platform based on SOA and Cloud Computing
    - Enabling automotive to work with greener society such as smart grid
- **Future Work**
  - Remaining some challenges: non-functionality aspect, safety-critical issues like QoS and dependability
  - Make ACSS more usable to the development of automotive software systems

Acknowledgements

- Special thanks for external collaborators:
  - Norio Oohashi, NEC Corporation
  - Osamu Takahashi, Eiwa System Management, Inc.
  - Teruyuki Nakazawa, DENSO CREATE Inc.
  - Steven Kelley, Metacase
  - Prof. Aoyama, Nanzan University

- Additional reference:

  - A. Iwai, et al., Experiences with Automotive Service Modeling, Proc. of 10th Workshop on Domain-Specific Modeling, ACM SPLASH 2010, Oct. 2010

DENSO

Thank you for your attention!

Prototype DENSO Electronic Vehicle

for commemorating the 60th anniversary of DENSO COPORPORATION