PSOS, Parking Space Optimization Service

Thomas B. Hodel-Widmer, University of Zurich
Suo Cong, University of Zurich

Conference paper STRC 2004

STRC 4th Swiss Transport Research Conference
Monte Verità / Ascona, March 25-26, 2004

PSOS, Parking Space Optimization Service

Thomas B. Hodel-Widmer, Suo Cong
University of Zurich
Department of Information Technology
Winterthurerstr. 190
CH 8057 Zürich
Switzerland

Phone: +41 1 635 4576
e-Mail: {hodel, suocong}@ifi.unizh.ch

Abstract

Amongst the wide range of parking solutions that can contribute to reduce parking problems or regulate parking activities, e Parking looks at developing and applying an innovative e-business application for parking space optimization.

The purpose of this paper is to present the innovative e-business platform that has been developed, from a technical point of view, by the University of Zurich. The ideas are coming from a transcross European consortium within the framework of the IST Information Society Technologies of the 5th framework program.

E-Parking provides a database-centered Web application solution based on our proposed conceptual model CIA (Channel, Integration, Application) for Web applications. The WAP, WEB and Bluetooth communication channels enable drivers to obtain early information on available parking space, make a reservation, access the reserved place and pay for the service booked. In reaching this goal, the innovative solutions seek to benefit all social segments, to optimize existing parking resources, and to contribute to achieving a more sustainable urban transport, reducing congestion and pollution.t

Keywords

E-Parking, Web Application, Web Service

1. Introduction

Parking activities constitute an extensive area that greatly affects overall mobility of car drivers as all vehicle trips require a parking space at the destination. The influence of parking activities in urban transport is multi-directional as an integrated component of the overall urban transport supply.

Related problems derive from the driver's need of affordable and convenient parking space in areas of scarce supply where drivers' search for parking spaces may account for up to 30% of urban traffic flows and a correspondingly high proportion of CO2 emissions.

On the other hand, excessive parking offers are also a source of problems since parking facilities are costly and affect all sectors of the community: government, developers, users, residents and nearby businesses. Besides the associated environmental costs, excessive offer contradicts often transport development strategies looking for sustainable mobility, since free cost parking promotes car use and discourage the use of other alternative modes.

The e-Parking project has focused on the development of a secure e-business tool for a user-friendly and efficient parking space service based on mobile access. The idea behind E-Parking is illustrated in figure 1, where a platform, referred to here as PSOS (Parking Space Optimization Service) is the gateway through which drivers can easily find parking offers matching their needs. As seen in figure 1, the PSOS is designed to serve all transport segments where a parking space is needed, in other words the PSOS acts as a Parking Brokerage Service.

Figure 1: The e-Parking model

The proposed tool has been integrated into an easily deployable system for parking suppliers. From the end-user perspective multi-channel access to the services is provided. The platform enables the incorporation of existing payment schemes,

including m-payment. In order to make all processes accessible from the mobile phone, access control is additionally provided via Bluetooth, allowing the user to interact with existing parking equipment when entering or leaving a car park and to be charged based on registered time stamps.

Benefits of e-Parking

End User	Parking Sector	Other business sectors
Reduced search traffic	Parking spaces increased in high demanded areas	Good access for cars ensured in their proximity
Easier payment at parking facilities	Parking occupation enhanced	Parking costs transferred to parking users
Parking availability ensured at trip destination	Anti – fraud measures improved	Revenues increased
Customized information service possible	Security increased (cashless payments, knowing the customer)	Mobile terminals, mobile ticketing is an important application
Pre-trip / On-trip parking information improved	Better management of parking operation	Network Operator
	Increased revenues from parking operation (optimi-zation of car park spaces, secured payments, customer loyalty, image: market innovator)	Wireless applications, the e-Parking application is a promising business appli-cation for mobile ticketing, which could be applied outside the parking domain as well
Sustainability of e-Parking: Economically, Ecologically, Socially		

1.1 User requirements and system functionality

The two target user groups of a Parking Brokerage Service are primarily drivers on the one side and parking space suppliers regardless of their type on the other side.

For the primary categories, the essential aspect that needs to be taken into account when specifying a travel related service are the information needs of the users. In general, drivers, as other type of travelers, expect services that provide accurate and real time information, meet their preferences and particular needs, and are available whilst being on the move.

Parking suppliers expect services that are adaptable or complementary to their parking management policy, can help them to improve existing services or that

can bring them the possibility to offer new ones to their customers, can optimize their parking management operations, and that are related to their parking location and the purpose of use of their customers.

While information and communication technologies constitute the building blocks of e-business platforms, particular attention has to be paid to the technological and information requirements that an integrated system may impose either on the driver's side or on the supplier's side. Thus the ability to deploy the system platform easily and ensuring the transparency of operations for the end user cannot be overlooked.

Based upon these considerations the essential requirements that have been extracted to respond to the user needs are summarized below.

Real-time information provision, which shall be precise and accessible by the users of the system, makes use of available communication infrastructure.

Flexible payment options shall be provided by the system including current credit or corporate cards schemes, pre or post paid payment as well as enabling other mobile payment options.

Ease of use characterized by a destination driven service including Geographical Information Service (GIS) modules, which is fast and easy to operate, and that is customizable permitting the introduction of user preferences and the information about services related to the parking destination (i.e. places of interest, events).

A highly interoperable platform is needed, which is adaptable to different parking suppliers and proprietary systems and which allows transparent integration of Bluetooth devices.

High optimization of operations shall be provided and in particular fast user recognition in relation to entry and exit times that is comparable to existing access control technology such as manual operation or use of parking cards, chip coins or other established technologies.

1.2 Innovative aspects

PSOS is an interactive electronic market place, which matches parking space demand with offer and handles payment flows to the payment service provider (financial institution, telecoms operator…). PSOS is accessed both by drivers through Web, SMS or WAP technology and by parking space providers through Web Services to the central database. Bluetooth technology then enables the driver to be recognized at the entry and exit points of the parking lot and triggers the secure m-payment procedure which debits his credit card or phone bill.

1.3 Overview of system functionality

1. The parking space provider offers parking space available for reservation. This information is registered in the PSOS database.

2. Users are able to access the PSOS via Internet or WAP for obtaining parking information or for making a reservation request. The reservation request is registered in the PSOS database.

3. The PSOS sends the booking information and access code to the end user subject to the acceptance of the reservation request by the Parking Space Provider.

4. The car enters and exits the parking facilities using Bluetooth to open the barrier.

5. Once the car exits the car park electronic payments is made and the whole operation is registered in the PSOS.

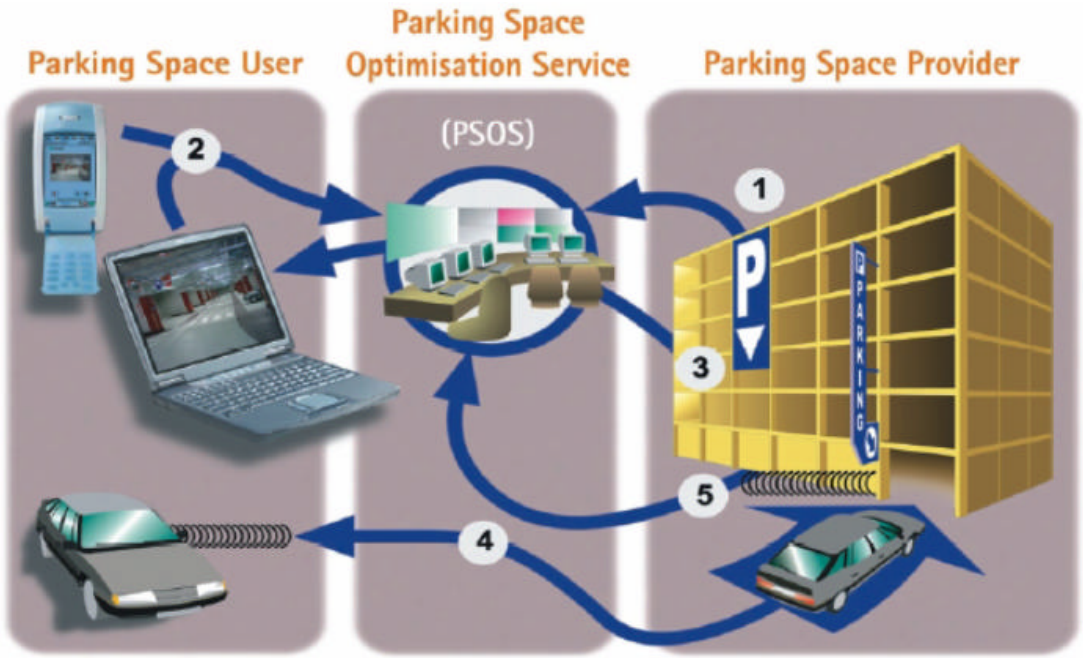

Figure 2: System functionality

PSOS offers services like, Administrative Services to make the PSOS system more configurable and flexible, ease the management and maintenance of PSOS system, Car Park Provider Services, to register car parks and provide the parking

spaces, manage the reservations via multiple channels (e.g. Internet, mobile devices…), facilitate the management of car parks, and End User (Customer) Services to ease the search and reservation of parking spaces.

The paper is organized as follows. We begin in section 2 by presenting the proposed conceptual model CIA (Channel, Integration, Application) for web applications. Section 3 describes the e-Parking system design and implementation. We conclude in section 4 and discuss the future work.

2. The General Conceptual Model for Web Applications

Accompanying with the rapid progress of Internet and the ubiquity of the World Wide Web, more and more applications are going to use the Web as the platform to deliver services and products to their subscribers. Meanwhile, many new kinds of applications or new ways to do legacy applications such as e-commerce are motivated by the innovative Internet/Web technologies.

The concept of Web applications has slightly different meanings. A common definition of Web application is to regard a Web application as a Web system that consists of web server, network, web browser and etc. The user's activities such as navigation and input data will affect the state of the business [Con99]. In this definition, the Web application is characterized by two features: 1) it is a software system with business state; 2) its front end is delivered mainly via a web system. The focus on the business logic and business state distinguishes a web application from a web site. Another definition is to define a Web application as a software application that depends on the Web for its correct execution [GG99]. In this definition, any software, which is designed to deliver over the web, is regarded as a web application. A web site is a kind of web applications because its content is delivered over the web.

To facilitate the design and development of Web applications, we need a suitable conceptual model or models and efficient developing tools/environment. Although many efforts have been devoted to design models of web-based applications, the development, engineering and management of such kind of applications are still not well supported ([Sch98], [GG99]). One reason may be the lack of a clear conceptual model for the overall architecture of web applications. The general architecture of most web applications is based on client/server architecture and N-Tiered architecture. A typical web application system is composed of database server, application server, and web server. The application logic is distributed and mingled across all tiers: web pages that contain client side /server side scripts, middleware, business applications, data access and integration. The absence of a clear conceptual model leads to ad hoc development of most web applications.

2.1 CIA Model

We define web applications in an evolutionary way. A software application is characterized as a web application if:

1) It uses web to deliver information/contents;

2) It uses web to deliver functionalities or e-services;

3) It uses web to compose and integrate business processes.

Our definition highlights the leverage of Web technologies (e.g. HTTP, HTML, Web Services, etc.) to develop web applications. The primary difference between various web applications is the extent to which the applications use web technologies. The web is not only a medium of exchanging information but also a rapid evolving execution platform / runtime environment for software applications from small scale to large scale. A web application is not only a structured set of web pages plus scripts.

In this paper we propose a general conceptual model of web applications. This model, which is named as CIA, consists of three primary constructs: Channel Manager, Integration Manager, and Application Coordinator. The overall application logic is partitioned into three isolated parts. The application logic that used to interact with users is aggregated in Channel Manager. The application logic that used to access data and to invoke other functionality outside the application or to be invoked by other applications is grouped in Integration Manager. The application logic that realizes the actual business functionalities is regarded as main application logic, which controlled by Application Coordinator.

Different web applications may put different emphases on those constructs. For example, a typical web site will consider the graphic user interface is more important. However, an e-commerce application will emphasize the integration of data and business processes among partners.

We will use our e-Parking application to illustrate the general conceptual model in the following.

2.2 Channel Manager

Naturally, one of the most remarkable features of Web applications is the navigational ability. So some proposed Web application models like HDM/HDM-lite ([FP00]), OOHDM ([SR95]) and RMM ([ISB95]) clearly separate the navigational aspects and structural aspects from visual aspects in order to gain a more effective and expressive design model ([BP00]). As an example, HDM-lite describes a Web application through three schemas: a structure schema, a navigation schema, and a presentation schema. Different approaches have been employed to emphasize different aspects of Web applications and focus on different standpoints ([GPS93]).

The CIA model extends this idea of separation by introducing the Channel Manager. The core of the Channel Manager is a device engine, which is used to generate different presentations in appropriate formats (e.g. HTML, WML) to different devices (e.g. PDA, cell phone, kiosk), to support different navigation patterns, and to receive user inputs from different devices. The device engine consists of a set of different navigation/presentation models for different devices and provides a uniform interface to the Application Coordinator. When a new kind of devices is added into the application system, the Channel Manager is responsible to create new navigation/presentation model (automatically or manually) for the new device. The main application logic, which is managed by Application Coordinator,

doesn't need to change if the interface between Channel Manager and Application Coordinator keeps intact.

The purpose of the device engine is to support multiple accesses to web application without changing the application logic.

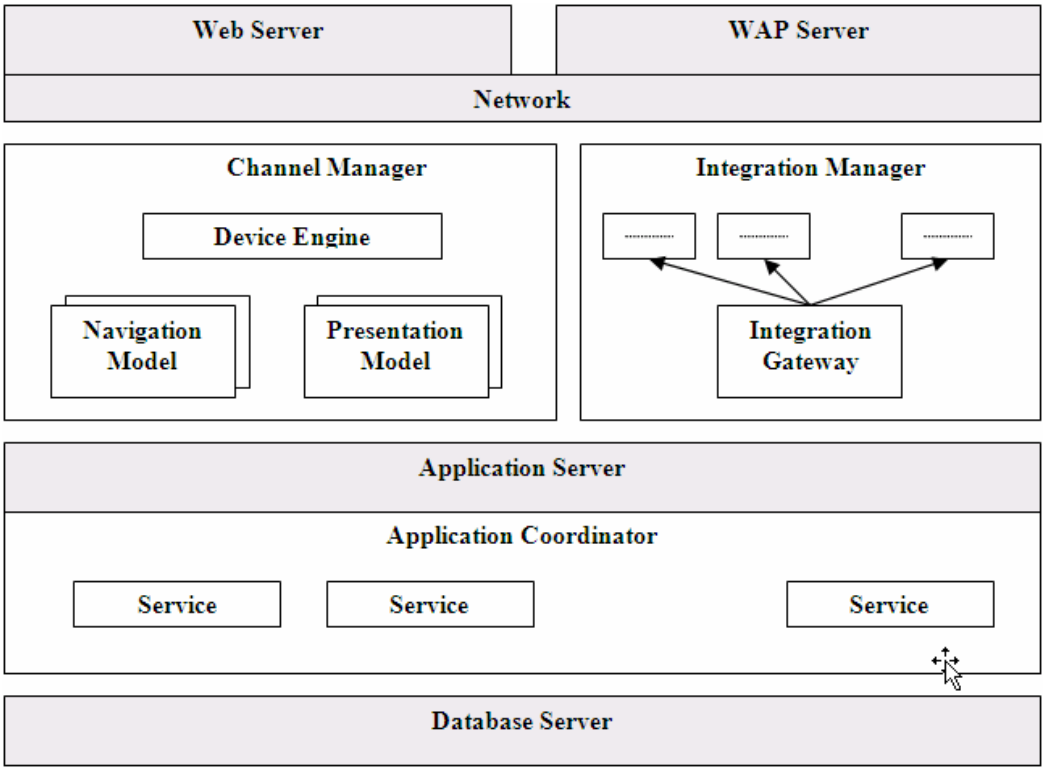

Figure 3: The CIA model

2.3 Integration Manager

It is increasingly difficult to partition one application system from others clearly for today's information systems [Has00]. The borders of applications across enterprises are blurring. One remarkable reason is that data required by applications is becoming more and more dispersed over heterogeneous and autonomous information systems. Another reason is the growing requirement and impetus of cooperation among business partners and their information infrastructure. For Web applications, the integration approach is even more crucial to achieve a successful application development due to the 'Web' nature of the application.

The system integration can be addressed in different layers ([Has00], [Sta02]). A three-layer integration approach, which includes inter-organizational processes, enterprise application integration and middleware integration, is discussed in [Has00]. Although the border between different layers can not always be separated definitely the 'divide and conquer' methodology leads to easier and efficient solutions. In this paper, we propose a novel two-dimension integration approach. The one axis is the hierarchical layers of the information systems. Another axis is

used to depict the extent to which the integration occurs inside the application or outside the application. On the one extreme, the inside integration means the application will assume all integration tasks. This is too cumbersome to design such an application. On the other extreme, the outside integration means the runtime environment (for web applications, it includes Web itself) of the application will assume all integration tasks and the integration is transparent to the application. This is obviously too optimistic. The first reason is that each application has its own specific requirements of integration. In addition, the objects (data, functionalities, processes), which will be integrated into an application, are dynamic and always changing even after the application is developed and deployed. For example, new data sources will be ready to access and exchange. It is also possible that the application is required to cooperate with new partner's business processes. So the integration is always a mixture of application inside integration and outside integration.

Web technologies, XML, middleware technologies (e.g. CORBA, COM, EJB) as well as the technology of component-based software provide supports for application outside integration. However it is not sufficient to simply combine the component-based middleware and Web technologies [Sta02].

Web Services provide a technology that enables companies to encapsulate applications and business processes for integration with multiple business partners. In other words, it enables a new way to compose web-based applications through Internet/Web. Because Web Services are completely based on Web standards and XML, the integration via Web Services is superior to other approaches such as proprietary EAI solutions [Sta02]. There is a wide spectrum of products supporting Web Services, such as database systems, application servers, and office suites. It is becoming an integral part of modern information system architectures [DM03].

We highlight the role of Integration Manager in our model that facilitates the design and maintenance of those Web applications. The key idea behind the integration manager is to separate the integration tasks from the main application logic. The integration manager is leveraged as a bridge between the runtime environment and the core of the application.

For example, in the e-Parking application system, PSOS will collect the necessary data (e.g. how many free parking spaces, prices for parking, opening time) from various car parks with different information infrastructure. Some car parks use DBMS systems to manage their data and some car parks use just structured files. Furthermore, car parks often provide different ways to access their data. Although Web Services solutions are apparently promising with many technical advantages, only a few car parks will prepare to support it so far. For those small car parks, which don't want to afford such an information system, PSOS has to provide them an interface (e.g. a web page) to input those data manually. One task of the Integration Manager is to release the main application logic from the 'physical' queries, which directly posed against the car parks. It provides a 'logical' query functionality, which serves as a dispatcher to those physical queries, to the main application logic. When a new car park joins the PSOS, the main application logic

doesn't need to change. The only thing needed to do is to create a new 'physical' query for the new car park automatically or manually in the Integration Manager.

Due to the fact that Web Services are still a fast moving and immature technology, interoperability between different implementations cannot be guaranteed [Sta02]. This is proved in our e-Parking applications. The PSOS provides Web Services via different channels, e.g. Web and WAP. Meanwhile, PSOS has to access the Web Services from car parks to query data, send requests of reservation and etc. Because different implementations of Web Services have been involved, the Integration Manager of PSOS has to construct an adaptor to achieve the desired interoperability.

2.4 Application Coordinator

The Application Coordinator is used to manage and control the main application logic, and to coordinate the main application logic with Channel Manager and Integration Manager. The main application logic is structured into functional components with different granularities based on application-specific model. The detail architectural model of the Application Coordinator is open.

Since database systems have played a very important role in Web applications ([CFP99], [FLM98], [Wie99]) in different aspects, we propose a database-centered pattern for the Application Coordinator. This pattern, which is powered by the matured DBMS technology, defines the border of the application based on the access right of the database. The main application logic involves all functional components that 1) can access the system database directly; 2) cannot access other databases directly. The access to other databases must be carried out by Integration Manager. And any components belonging to main application logic must be invoked via Integration Manager. For example, in the e-Parking application system, the PSOS function 'makeReservation' can access the PSOS database directly. So it belongs to the main application logic. So when a driver makes a reservation, the request is dispatched to the Integration Manager whether it is via Web or via WAP. The Integration Manager will deal with the differences in the process of accepting a reservation at different car parks and use different ways to send the request. The function 'makeReservation' doesn't need to know the details of how the request is sent and just concentrate on the functionality of making a reservation.

3. System design and implementation

Technically, the project rests on an integrated architecture composing a WAP cellular phone or a standard internet access and a Parking Space Optimization Service (PSOS) interfaced with the parking space provider's reservation system.

Due to the wide diversity of user terminals and operation systems the platform mostly runs on generic technologies, namely WAP, an open specification that enables mobile users to interact with services using a wireless device, and the Bluetooth protocol, a wireless communication technology providing a short distance radio link between devices such as computers or mobile phones. All interfaces are realized with our unified web services architecture.

One of the core elements of the system is the PSOS, in effect an interactive electronic market place, which matches parking space demand wit offer. PSOS goes beyond existing parking reservation systems, introducing a new form of parking space management by, offering the possibility of pooling parking space.

PSOS is accessed both by drivers through WEB, WAP technology and by parking space providers through a specific interface to the central database. Bluetooth technology then enables the driver to be recognized at the entry and exit points of the parking lot. The system aims at supporting different business models in which payment flows can be either handled through the PSOS platform or at parking provider level.

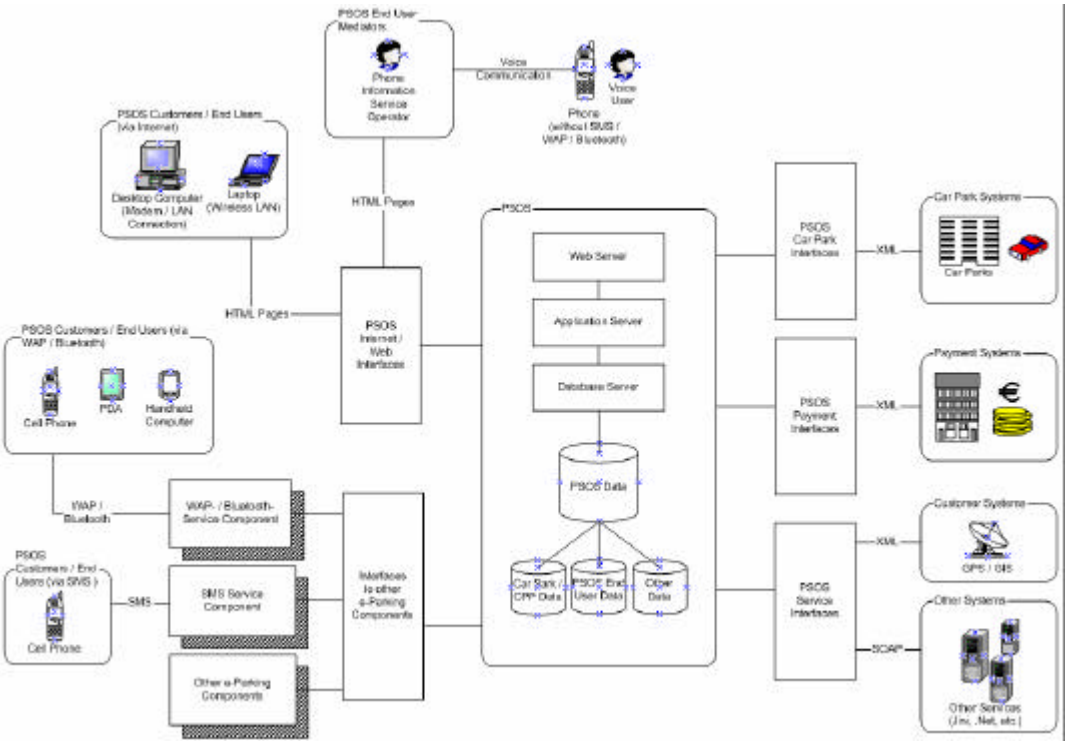

Figure 4: Software architecture

3.1 Overall system architecture

This specification describes the organization structure of PSOS database that is implemented within Cache[1], a post-relational database. The 'post-relational' refers to the feature that Cache goes beyond the limits of relational databases while still supports a standard relational view of data.

Firstly, Cache is a full-featured relational database. All the data within a Cache database is available as true relational tables and can be queried and modified using standard SQL via ODBC, JDBC, or object methods.

On the other hand, Cache provides the ability to model data as objects each with an automatically created and synchronized native relational representation. So the database applications can benefit from the object-oriented technology such as inheritance, polymorphism and etc. Cache supports to intermix SQL and object-based access within a single application, using each for what they are best suited.

Cache provides two distinct types of classes: data type classes and object classes. The data type class is used to represent literals such as strings, integers, and other types defined by the user (designer or programmer). The object class defines the data structure and the behavior of objects of a type. These objects are called instances of their associated class and the creation of an object is called instantiation.

PSOS database is based on Cache object model and each entity is modeled as a persistent object class or an embeddable object class.

[1] http://www.intersystems.com

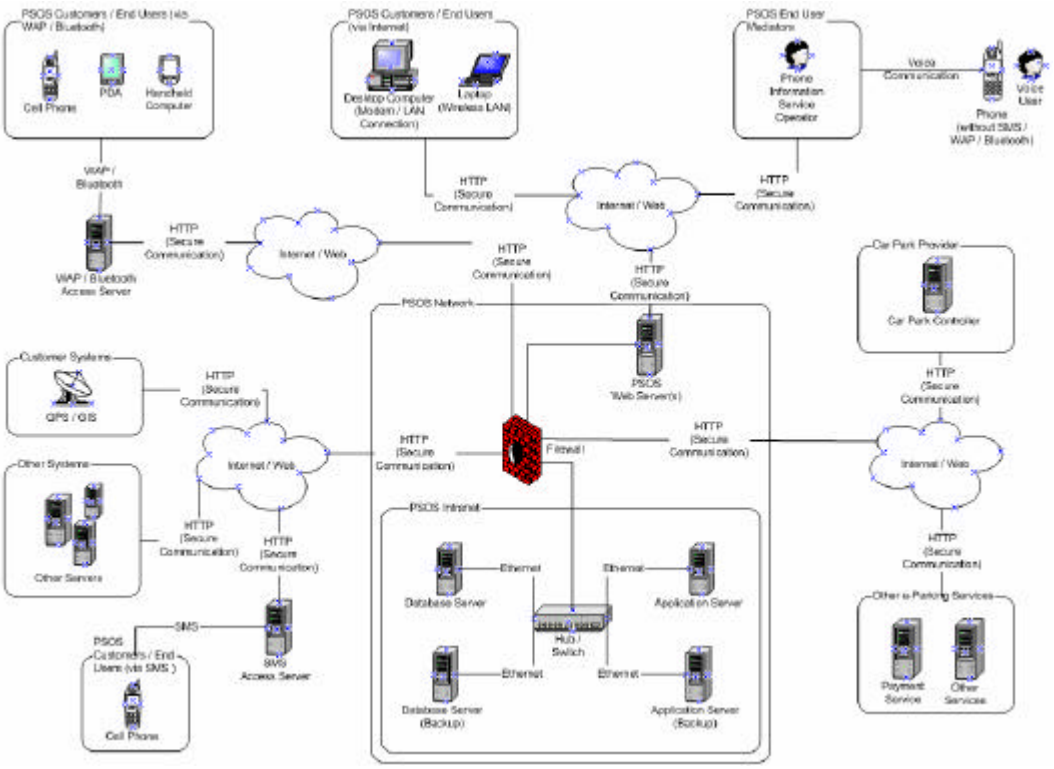

Figure 5: Hardware architecture

Persistent Object Classes: Simply, persistent object classes have a comprehensive behavior for the permanent storage of its instances (the objects) in the databases. Persistent objects have an object identity and a unique object identifier (OID). Each persistent object can be stored independently in Cache.

Embeddable Object Classes: The embedded objects cannot be stored independently in Cache; they must be stored in database only embedded in other persistent object classes. For example, in PSOS, 'Address' is defined as embeddable object class (Serial Class) that is used in class 'CarParkProvider', 'CarPark' and 'Customer'.

Car Park Provider: The class 'CarParkProvider' is used to model a car park provider who has one or more car parks registered in PSOS database. The relation between 'car park provider' and 'car park' could be one to one (one car park provider, one car park) or one to many (a car park provider could have several car parks registered in PSOS) if the policy allows that. This will be determined by the E-Parking policy in the future.

Car Park: The class 'CarPark' is used to model a car park that is registered in PSOS. Class 'CarPark' has a reference property that refers to another class 'CarParkProvider' that specifies the provider (owner, manger or administrator) of the car park.

The property 'IsPSOSDependent' is used to specify the relation of operations between PSOS and the registered car park. Currently PSOS database support two kinds of car parks: dependent car parks and independent car parks.

If a car park is dependent on PSOS (IsPSOSDependent = true), the PSOS is in charge of the normal operations such as make a reservation, or cancel a reservation. The provider of a dependent car park could log in PSOS to manage his car park. The application of this feature is still under discussion because it will involve many management and legal issues in implementing.

If a car park is independent on PSOS (IsPSOSDependent = false), the PSOS will send the service requests (e.g. make a reservation) to the car park and wait the responses from the car park for further process. In this case, the independent car park should configure its registration profile to specify related services opened to PSOS, for example, the web service of make a reservation.

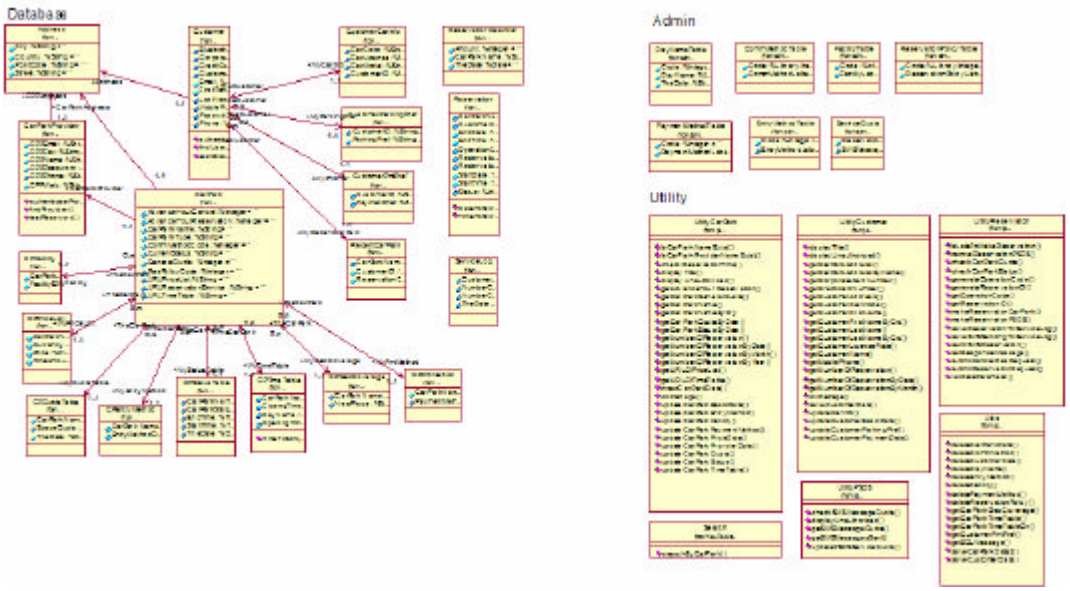

Figure 6: The PSOS data schema

The class 'CarPark' has several properties of relationship, which are used to model an association between a 'CarPark' object and another object (or table in relational view). The class 'CarPark' models the most basic features of a car park. Other features, which are more dynamical and complicated, are modeled by a set of associated classes: CPGeoCoverage, CPTimeTable, CPStatusTable, CPPriceList, CPPmtMethod, CPEntry, and CPFacility. This structure makes PSOS database more flexible and easier to extend its functionalities.

Address: The class 'Address' is used to model the address information in PSOS database. It defines a common format to describe an address. It is a serial class that embedded in other classes.

Car Park Geographical Coverage: The class 'CPGeoCoverage' is used to model the geographical places covered by the car park. The property 'NearPlace' is used to specify a place that is near to the car park. A car park can specify multiple places that are nearby. The address of the car park (the street at which the car park locates) is default set by PSOS as a 'NearPlace' of the car park.

Car Park Time Table: This class 'CPTimeTable' is used to model the opening time of a car park. The property 'DayName' specifies the name of the day that defines an opening time (from) and a closing time (to). By far, the 'DayName' has a value set from Monday to Sunday. However, this structure is not limited to describe timetable only for normal days. It provides the capacity of extending to support complicated timetable in the future. If complicated timetable involves other special days such as public holidays, a new table could be introduced to define a mapping between the name of the special day and the exact date.

Car Park Status Table: The class 'CPStatusTable' is used to model the status of a car park according to time. The status of a car park represents the its availability. If the status of a car park is 'red', then this car park cannot accept any reservation (e.g. it is almost fully occupied or the car park is closed for maintenance). But if the status of a car park is 'green', then this car park can accept reservations and it must have enough free parking spaces. In PSOS database, only 'red' status will be recorded to save spaces. The property 'Status' is not necessary in current stage because only 'red' status will be recorded in database. However, we keep this property for future extension.

Car Park Quota Table: The class 'CPQuotaTable' is used to model the quota table of parking spaces of a car park according the date. A quota is the number of available parking spaces (parking lots) on the specified date, which is guaranteed by the car park.

Car Park Price List: The class 'CPPriceList' is used to model the price list of a car park.

Car Park Payment Method: The class 'CPPmtMethod' is used to model the payment methods that supported by a car park. The codes of payment methods are defined in another class 'PaymentMethodTable'. The class of 'PaymentMethodTable' is designed to provide a central way of managing payment methods that offers flexibilities for modifying the label of a payment method and supports future extension of introducing new payment methods.

Car Park Entry Method: The class 'CPEntryMethod' is used to model the entry methods supported by a car park. The codes of entry methods are defined in another class 'EntryMethodTable'. The class of EntryMethodTable is designed to provide a central way of managing entry methods of car parks. The label of one existing entry method can be changed easily and new methods can be added into PSOS database on demand because the entry methods of the car park are stored as codes in database and the mapping between a code and its label is defined in the class/ table 'EntryMethodTable'.

Car Park Facility: The class 'CPFacility' is used to model the facilities provided by a car park.

The codes of car park facilities are defined in another class 'FacilityTable'. The class of FacilityTable is designed to provide a central way of managing facilities of car parks.

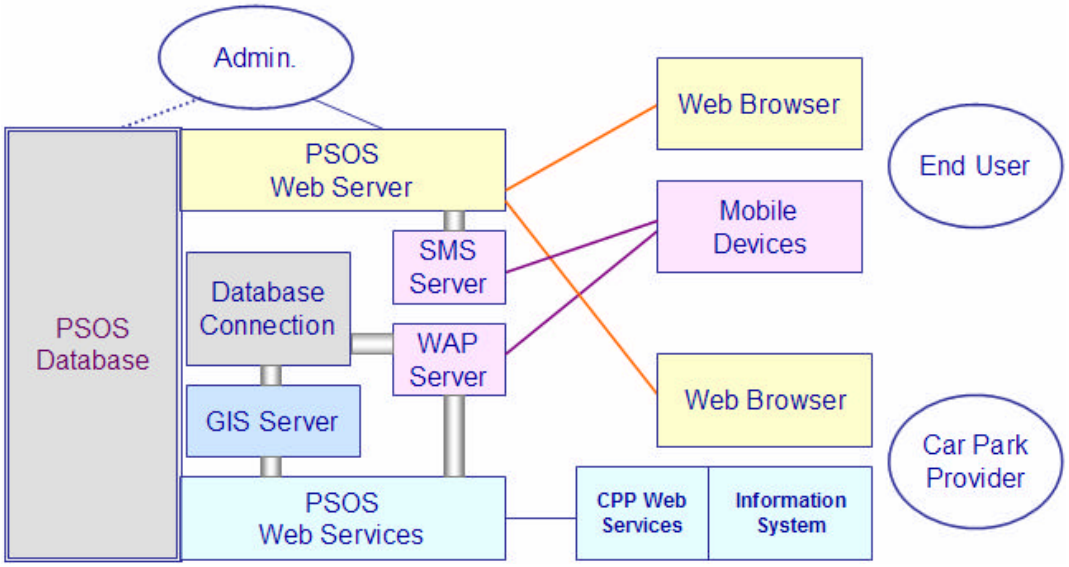

Figure 7: PSOS Block Diagram

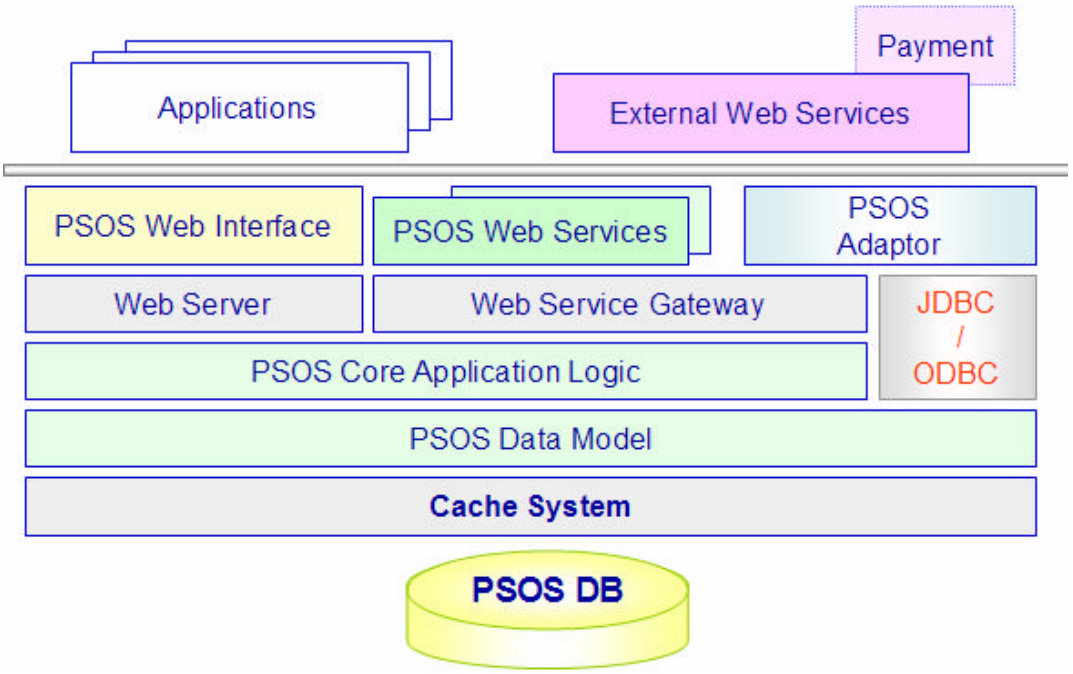

Figure 8: Hierarchical Architecture

Customer: The class 'Customer' is used to model an end user (driver) who registers in PSOS system.

Several properties of relationship are defined to describe different aspects of information about the end user: CusotmerCarInfo, CustomerPmtPref, CustomerParkingPref, and RecentCarPark.

Customer Car Information: The class 'CustomerCarInfo' is used to model the information of the customer's car.

Customer Payment Preference: The class 'CustomerPmtPref' is used to model the payment preferences of the customer.

Customer Parking Preference: The class 'CustomerParkingPref' is used to model the parking preferences of the customer.

Recent Car Park: The class 'RecentCarPark' is used to model the car parks at which the customer made a reservation recently. It is a 'system generated' object (table) that produced by the PSOS system according to the reservation requests submitted by customers.

Reservation: The class 'Reservation' is used to model the reservation request of a customer at a car park.

Reservation Calendar: The class 'ReservationCalendar' is used to model a calendar of reservations at a car park. It specifies how many reservations are made at a car park according to the date. For a multiple-day reservation, each day inside the duration of the reservation will be recorded.

4. Conclusion

In this paper, we propose a general conceptual model, the CIA model, of Web applications. The primary methodology embedded CIA model is 'divide and conquer'. It anatomizes a Web application with 3 basic constructs: Channel Manager, Integration Manager and Application Coordinator and provides a systematic way of organizing the whole application logic that will distributed over database server, web server, application server and so on. It separates the main application logic from logic related to presentation/navigation via multiple channels and from logic related to system integration. All the three kinds of logic are very important but different applications may lay different emphasis on different constructs. We illustrate the CIA model by presenting e-Parking PSOS application, which is an innovative e-business platform. In the context of e-Parking application, we propose a database-centered pattern for Application Coordinator in the CIA model. The leading purpose is to apply the matured and state-of-the-art database technologies to design Web applications in a comprehensive manner. The PSOS has been integrated into an easily deployable system for parking suppliers. From the end-user perspective multi-channel access to the services is provided. The conceptual model behind PSOS enables the incorporation of existing payment schemes, including m-payment. In order to make all processes accessible from the mobile phone, access control is additionally provided via Bluetooth, allowing the user to interact with existing parking equipment when entering or leaving a car park and to be charged based on registered time stamps.

5. References

[BP00] M. Bochicchio, and R. Paiano. Prototyping Web applications, Symposium on Applied Computing, Proceedings of the 2000 ACM symposium on Applied computing 2000.

[CFP99] S. Ceri, P. Fraternali, and S. Paraboschi. Design Principles for Data-Intensive Web Sites. SIGMOD Record, 28(1), March 1999.

[Con99] J. Conallen. Modeling Web Application Architectures with UML. Communications of the ACM, 42(10), 1999

[DM03] B. Daum, and U. Merten. System architecture with XML. Morgan Kaufmann Publishers, San Francisco, CA, 2003.

[FLM98] D. Florescu, A. Levy, and A. Mendelzon. Database techniques for the world-wide-web: A survey. SIGMOD Record, 27(3), 1998.

[FP00] P. Fraternali, and P. Paolini. Model-Driven Development of Web Applications: The Autoweb System. ACM Transactions on Information Systems, 28(4), October 2000.

[GG99] H.-W. Gellersen, M. Gaedke. Object-Oriented Web Application Development, IEEE Internet Computing, 3(1), Jan.-Feb. 1999.

[GPS93] F. Garzotto, P. Paolini, and D. Schwabe, HDM - A Model-Based Approach to Hypertext Application Design, ACM Transactions on Information Systems, 11(1), January 1993.