A WEB-BASED AUCTION DESIGN FOR MASS CUSTOMIZATION OF SERVICES

NIMA ESLAMLOO

A THESIS

IN

THE DEPARTMENT

OF

CONCORDIA INSTITUTE FOR INFORMATION SYSTEMS ENGINEERING (CIISE)

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF APPLIED SCIENCE IN

QUALITY SYSTEMS ENGINEERING AT

CONCORDIA UNIVERSITY

MONTRÉAL, QUÉBEC, CANADA

JULY 2013

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Nima Eslamloo**

Entitled: **A Web-Based Auction Design for Mass Customization of Services**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science in Quality Systems Engineering

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Andrea Schiffauerova	Chair
Dr. Yuhong Yan	Examiner
Dr. Jamal Bentahar	Examiner
Dr. Chun Wang	Supervisor

Approved by ______________________________

Chair of Department or Graduate Program Director

2013

Dr. Robin A. L. Drew, Dean of the Faculty of Engineering and Computer Science

Abstract

Mass customization provides customers with the ability to design products and services according to their individual needs through highly flexible processes. In the context of services, this approach calls for the effective allocation of limited service capacity in order to meet customer requirements, thereby increases customers’ value on a product in terms of its available options, price, and schedule.

In this thesis, we introduce a web-based auction design for the mass customization of services under capacity constraints. The proposed system design integrates customers’ decision making with a decentralized service customization process through a web-based auction model. This web-based auction system is implemented using an iterative bidding procedure in order to maximize the overall customer value given limited capacity. Experimental results indicate that the solutions obtained from our web-based auction closely approximate those of the optimal outcome. Moreover, it was found that reductions to services customizability significantly decreased customer overall value and auction revenue.

Keywords

Mass customization · Capacity allocation · Service customization · Combinatorial iterative auction · Web services · Web-based auctions

Acknowledgments

The fulfillment of this work would not have been possible without the help and support of the kind people around me.

Foremost, I would like to express the deepest appreciation to my supervisor, Dr. Chun Wang for the continuous support of my master's study and research, for his patience, motivation, enthusiasm, and immense knowledge. Without his guidance and persistent help this thesis would not have materialized.

I owe my deepest gratitude and appreciation to my family and beloved partner Bahar Nahjavi for their continuous support and encouragement. No words can express how grateful I am for your love and support, and how much I love and appreciate you.

I would also like to thank Farnaz Dargahi for offering me her kind guidance, suggestions, and support throughout this research. Special thanks go to Shervin Nowroozi for his valuable comments and feedback.

Last, but by no means least, I would like to convey my thanks to my other friends who provided a supportive and friendly environment to me during the past two years.

Table of Contents

List of Figures

List of Tables

List of Acronyms

CSS	Cascading Style Sheet
DAO	Dara Access Object
GUI	Graphical User Interface
GVA	Generalized Vickery Auction
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IPO	Input Process Output
J2EE	Java 2 Enterprise Edition
JSP	Java Server Pages
OPL	Optimization Programming Language
PFA	Product Family Architecture
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
UML	Unified Modeling Language
XML	Extensible Markup Language

Chapter 1

Introduction

1.1 Background

This thesis is concerned with the development of a web-based iterative auction system for the mass customization of services. In general, mass customization can be understood as the ability to provide customized products or services via a flexible procedure in high quantities and at reasonably low costs (Da Silveira, Borenstein, & Fogliatto, 2001). Studies such as Fiore, Lee, & Kunz (2003) and Salvador, de Holan, & Piller (2009) indicate that mass customization provides an important competitive advantage in many economic sectors, including automobile and computer manufacturing. During the past decade, there have been significant developments in the mass customization of the product and service industries in particular, including applications in the web-based configurations, rapid manufacturing technologies, and more structured customer-interaction methods.

In a general sense, mass customization can be considered as a collaborative optimization process in which a company and its customers aim to find the best match between the customers' needs and the company's capabilities to meet these needs. In manufacturing customization, a company's core capabilities are the basis of its product families and their successive platforms (Meyer & Utterback, 1993). These capabilities are important because they are reflected in the people and properties used to develop new products. In order to support service customization, a product family architecture (PFA) is needed to distinguish individual customer needs and then gradually accomplish these needs by configuring and modifying well-established components and modules (Jiao & Tseng, 1999). While PFA enables customers to define their own requirements on the basis of a company's capabilities, these capacities can also be organized and illustrated using scalable product family design (Simpson et al., 2005) and configurational product family design (Du et al., 2001; Ulrich, 1995).

Scalable product family design well serves its customers in that the product platform can be stretched in many dimensions, based on customer needs. On the other hand, configurational product family design takes this concept further, delivering a modular product platform wherein product family members can add, substitute, or remove one or more functional modules in accordance with consumer needs.

Differently from existing literature that mainly focuses on customization in manufacturing, this thesis proposes a mechanism for the mass customization of services. The proposed customization mechanism is implemented by a web-based auction system using web service technologies.

Web service, which is a fast growing technology in the IT industry, has been inherited from Service Oriented Architecture (SOA). SOA is a paradigm by which software is designed to provide services through a standard interface, allowing services to be called from and used by other applications or software (Papazoglu, 2008). SOAs support a variety of different services from simple ones that process single tasks to complicated ones that perform complex business processes. Services that are implemented and used via the Internet are called web services. As stated by Papazoglu (2008), "A web service is a self-describing, self-contained software module available via a network, such as Internet, which completes tasks, solves problems, or conducts transactions on behalf of a user or application". Web services thus serve to facilitate interactions across other applications and services and to exchange information over a network. This research introduces a web-based auction design for the mass customization of services under capacity constraint. This model integrates customers' decision-making about service customizations within a company's capacity allocations through a web-based auction model networked between the company and its customers.

1.2 Motivation and Approach

To motivate our research from a practical perspective and to clearly demonstrate the application of our approach to real world problems, this thesis focuses its attention on one specific industry: online travel booking. This market has long attracted the attention of many people and organizations worldwide. Many of the online travel websites such as, ebay.com, luxurylink.com, and orbitz.com provide their customers with the ability to customize their own travel packages, typically through a "Build Your Own Package" service. However, different from our web-based auction system, the customization level in existing online travel auctions is really limited and they do not provide the flexibility of bidding on a customized package. Also, there are plenty of pre-packaged vacations offered by the aforementioned websites and other travel brands. However, as consumer travel wants and experiences are very unique and personal, customized packages are more attractive to the vast majority of consumers. As an example, ebay.com and luxurylink.com provide pre-packaged travel vacations to customers and they enable customers to submit bid on only that single package. If a customer wishes, he or she can look up on other packages and submit bid on that different package. However, in our web-based auction system, customers can customize several travel packages by choosing several travel services from a group of services and submit bit on their customized packages. A customized vacation package usually includes the following modules: flight tickets, accommodation reservations, entertainment tickets and/or car rentals. However for a special destination or within a special time window such as high-seasons, when the travel services of service provider are highly demanded, the capacity limited of the mentioned services restrict customers' options and affect the customizability. In this situation, the proposed approach can be highly profitable for both the service provider and also customers. The service provider can sell its travel services to customers who want to compete more and obtain a package with a lower and reasonable price. Also customers will have profit since they will buy and obtain a package including several items with a lower price from one place rather than buying the same items individually from different places. This thesis is driven by the following research question: "Having limited capacity, how can a service provider maximize the value provided to its customers by providing different levels of customization?" In the given scenario, the main objective of this approach is to maximize customized product values for a large group of customers. In terms of economics, such objective is called maximizing *social welfare* (Mass-Colell, Whinstom, & Green, 1995). In this study, service customization under capacity constraints is modeled as an optimization problem and a *design-by-customers* approach will be implemented using an iterative combinatorial auction. The auction serves as a collaborative framework that allows customers to participate in auctions and place bids on different travel packages through web-based applications. The main reason of implementing an iterative combinatorial auction in the proposed approach is because, bidders are better able to fully express their individual preferences when items, or in our case travel services, are complement. This means that, to a customer, a combination of services such as, a return ticket, a hotel and/or some entertainment tickets are more valuable than a single travel service. As an example, a pair of shoes have more value to a person than the value of each pair of shoe.

1.3 Outline of the Thesis

The rest of the thesis is organized as follows. In Chapter 2, we present a brief review of mass customization of services, various auction types, and web services technology. Chapter 3 formulates the service customization problem model and describes an auction model for mass customization in the travel industry. In Chapter 4, we analyse the design requirements of our web-based auction system. Chapter 5 presents the design and implementation of our web-based auction system. In Chapter 6, the performance of the web-based auction is tested and the experiment results are presented. Chapter 7 summarizes the thesis and presents future research directions.

Chapter 2

Literature Review

In order to contextualize the proposed web-based auction system in which customers will be able to customize travel service packages and bid on their customized packages, the term Mass Customization of Services must first be reviewed. Subsequently, a brief description of different auction models is presented. This chapter then concludes with an introduction to web services.

2.1 Mass Customization of Services

The ability to provide customers with a product or a service that is specifically tailored to their needs is a very valuable capability in today's industry. Mass customization is the means through which this goal has been pursued, whereby a producer is able to provide individually designed products and services to customers through a process of significant agility, flexibility, and integration (Davis, 1987).

For the purposes of this thesis, our focus is limited to the capacity aspect of mass customization wherein capacity is understood as the company's capability to provide a set of customized products during a predefined time schedule to a group of customers. In such a framework, it is necessary for a company to consider its capacity constraints in customization decision making, especially when production schedules are important for consumers. Furthermore, capacity constraints exert considerable influence on customer's satisfaction as well as company's revenue. Therefore, capacity constraints should be a key feature of service customization decision making.

In a manufacturing environment, mass customization's capacity constraints are normally directed from the customization management perspective and related to the direction of manufacturing planning and scheduling. Mass customization manufacturing aims to produce and control a variety of production planning and scheduling by using more flexible distributed coordination models for allocating resources (Tseng & Jiao, 2001). Instances of distributed coordination models can be found in holonic manufacturing (Guo, Hasegawa, Luh, Tamura, & Oblak, 1994), holonic-based architecture for process manufacturing (Chockshi & McFarlane, 2008), and agent-based manufacturing (Shen, Wang, & Qi, 2006).

There are available algorithms that can estimate resource availability in real-time, thereby supporting these distributed coordination models with their high levels of responsiveness (Moses, Gruenwald, & Dadachanji, 2008). For the purposes of manufacturing planning and scheduling, however, capacity allocation is not as great a concern as customers' requirements and negotiations. Instead, this factor is usually considered as a manufacturing issue. Service customization studies and research on the whole are limited in comparison to those of manufacturing environments (Da Silveira, Borenstein, & Fogliatto, 2001). Additionally, mass customization in service operations is one of the main omissions of the current mass customization literature.

In this understudied field of service customization, two forms of service operations can be distinguished: combinatorial and menu driven (Sampson, 2001). Combinatorial services operations consist of a combination of a group of services that are created as a unique service. Menu driven service operations, on the other hand, are formed through customers selecting a number of available options on the basis of their individual wants. The root of combinatorial and menu driven forms of service customization is the term *modularity*: the customization of various modules by customers. The customization model in this thesis is categorized as a design-by-customers model. Two phases of this model have previously been developed by Tseng & Du (1998): that of product design and that of customer needs acquisition.

In the product design phase, an iterative procedure exists by which customers are able to modify the attributes of a product through the customization of available services. This procedure is aimed so that customers can gain satisfaction from their customized product by the help of service configurations. Later, in the customer needs acquisition phase, customers are made aware of given design options by the service provider. The customers will then be asked to determine their desired product or service configurations according to what they have defined as valuable for each product.

Different from Tseng & Du (1998), the focus of this thesis is rather on the integration of a company's service capacity with the level of its customizability. It is assumed that a company's availability of products and services are given, and that customers know their ideal value for each of their selected and customized products or services. It is this service aspect of the customization model that will be further developed by this thesis.

2.2 Auctions

As this thesis proposes an auction-based service allocation method, we review some auction models in this section. The use and value of an auction system for the allocation of products and services has been well-known for centuries. Many different services are sold by auctions today, including flight tickets, museum tickets, concert tickets, temporary accommodations, and much more. A full list of items sold by auction is given by Cassady (1967).

Müller (2011) classifies auctions into two main categories: that of common-value auctions and that of private-value auctions. The former are auctions in which the value of the item being auctioned is the same for every bidder. However, each bidder will bring different estimates about the underlying value of the item up for auction. In private-value auctions, conversely, bidders will know the personal value of the item being auctioned, but may not have information about other bidders' values.

Service allocation literature further details many possible types of auctions, including single-object auctions, multi-unit auctions, Generalized Vickery Auctions (GVA), combinatorial auctions, and iterative combinatorial auctions. Each of the addressed auction types are reviewed as follow:

2.2.1 Single-object auctions

This type of auction is useful for settings where a single unit of an item is bought or sold one at a time. Examples of single-object auctions are Dutch auctions, First-price and Vickery auctions, and the most popular form of single-object auction: English or ascending-price auctions (Menesez & Monteiro, 2005).

The computation of single-object auctions is negligible. Nevertheless, these forms of auctions are widely used among auctioneers in the world.

2.2.2 Multi-unit auctions

Multi-unit auctions are usually used when a set or group of identical items need to be bought or sold together, rather than launching separate auctions for each item individually. If this is not the case and items are auctioned individually, such as in the case of single-object auctions, each item may be sold at different prices. This creates a “lumpy” bid problem (Tenorio, 1993) which multi-unit auctions would otherwise avoid. Multi-unit auctions can therefore be seen as a promising mechanism to allocate or re-allocate partible resources such as electricity generation and nature conservation contracts or the buyback of water rights in river environments (Hailu & Thoyer, 2006).

2.2.3 Generalized Vickery Auction (GVA)

In situations where bidders have pure private values, the Vickery auction (Vickery, 1961) provides an optimal mechanism to allocate a group of identical objects efficiently. However, in situations where bidders have independent values, the Vickery auction does not produce optimal efficiency. In such cases, the Generalized Vickery Auction can increase efficiency even if bidder values are independent from the values they reported to the auctioneer. The auctioneer will then be able to allocate its resources to bidders based on received values from the bidders.

2.2.4 Combinatorial auctions

Generally speaking, combinatorial auctions can be understood as important classes of market mechanisms in which bidders are allowed to bid on multiple heterogeneous resources that are bundled into packages (Narumanchi & Vidal, 2006). This type of auction is usually used in situations where participants have complementary values or similar financial constraints. An obvious advantage to combinatorial auctions is that bidders do not need to participate in multiple negotiations with providers for each individual item. Cramton (2006), moreover, points out the additional personal advantages of combinatorial auctions in that a bidder is better able to fully express his individual preferences in this format. This is very important when items are complements. An item can be complement when a set of items has greater value that a single item. A pair of shoes, for example, has more value than the left shoe alone. There are numerous examples of combinatorial auctions in practice. Computer science also studies the expressiveness of many bidding languages and the algorithmic aspects of the combinatorial problems. Consequently, much of the study on combinatorial auctions lies at the intersections of operations research, economics, and computer science (Cramton, Shoham, & Steinberg, 2006).

2.2.5 Iterative combinatorial auction

Iterative combinatorial auctions allow for bidders to submit multiple bids on bundles of items and for service providers to increase prices and maintain a provisional allocation in each round of auction procedures. In an iterative combinatorial auction participants can adjust their bids in response to bids from other participants and as the auctioneer updates provisional allocations and package prices. Although combinatorial auctions can be roughly approximated through multiple auctions on single items, this often results in inefficient outcomes (Bykowsky, Cull, & Ledyard, 2000).

The theory and practice of iterative combinatorial auction is well described in Parkes & Ungar (2000). Typical examples of iterative combinatorial auction are charted by Parkes & Kalagnanam (2005). A comprehensive survey of combinatorial auctions has been undertaken by deVries & Vohra(2003).

2.3 Web Services

As we will propose a web-based auction system which consumes travel services such as flights, hotels, and tickets from three different web services, it is important to review the concept of web services in general. The differences between web services and web-based applications will be shown, and finally the nature of a Service Oriented Architecture (SOA) will be discussed.

2.3.1 What is a Web Service?

The term web services is used to describe the ways in which services can be called on and used in a network. As defined by the World Wide Web Consortium, " A Web service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A Web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols," (W3C, 2004). Web services can provide various types of functionalities from simple requests to complex business processes. Funds withdrawal or funds deposits, weather reports, credit checking, and inventory status checking are some of the many examples of web services today.

2.3.2 Web Services versus Web-based Applications

An overview of web services would not be complete without a review of the differences between web services and web-based applications. These concepts can be principally distinguished in that web-based applications are normally developed and implemented to be used by humans, whereas web services are developed and implemented to be used mainly by machines. Additionally, web services do not necessarily have a Graphical User Interface (GUI) since they are typically used as a component in a larger framework. However, web-applications, as a complete framework, always have a GUI. Papazoglu (2008) further describes the characteristics of web services that serve to differentiate them from web-based applications as follows:

1) Web services act as resources to other applications with or without human intervention. This means that web services can be outsourced to other web services.
2) Web services are self-describing and modular. A web service can know what input it requires and what functions it can perform.

3) Compared to web applications, web services are far more manageable because the state of a web service can be monitored via external application management.
4) Web services may be auctioned. If multiple web services perform the same task, other applications can submit bids for the opportunity to use the requested service.

2.3.3 Service Oriented Architecture (SOA)

Service Oriented Architecture (SOA) introduces a set of design principles in which services with software applications or other web services communicate through a network by publishing interfaces. SOA mainly aims to produce an environment in which services and technologies can increasingly inter-operate. SOA and web services are two different, though highly related subjects. A SOA may be implemented without web services, though its deployment is made much easier through web services (Papazoglu, 2008). Generally speaking, SOA consists of three main roles: web service provider, web service registry and web service consumer (client) and three main operations: publish, find, and bind.

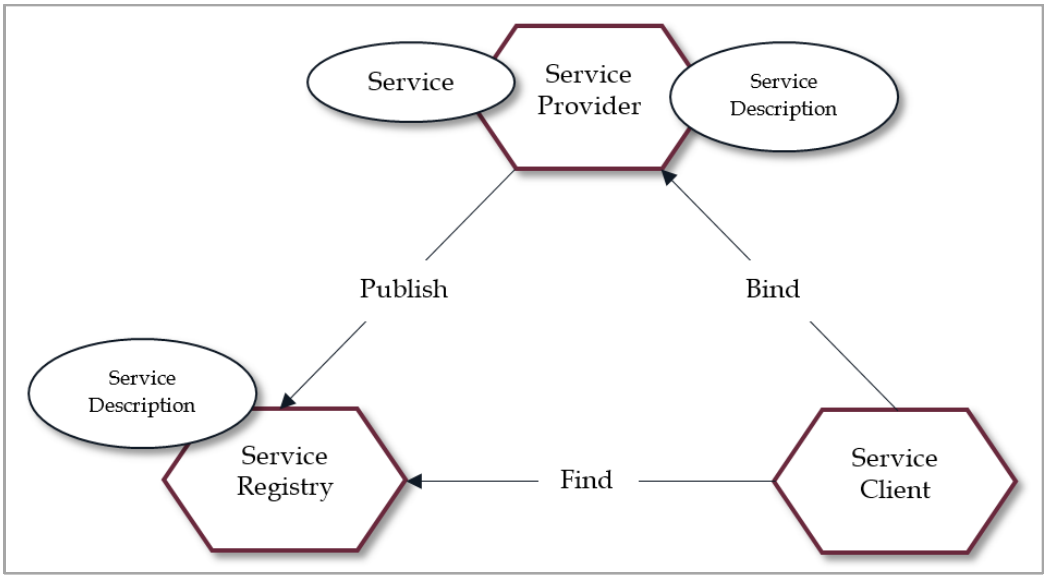

Figure 2-1: Service Oriented Architecture

Figure 2-1 depicts a typical Service Oriented Architecture, its main roles and operations. If a SOA is implemented using web services, it will consist of the following additional elements (Papazoglu, 2008, pp. 23-26):

- **Web Service Provider:** A web service provider is an organization who owns the Web service and implements the business logic for that Web service. Additionally, the Web service provider is responsible for *publishing* its Web services in a service registry which is hosted by a service discovery agency.
- **Web Service Consumer:** The web service consumer or client, is an enterprise who looks for an available Web service based on its individual

requirements. In order to find a desired Web service, the client must conduct a search in the service registry. If the enterprise finds an appropriate Web service it will *bind* to it.

- **Web Service Registry:** The web service registry is a searchable directory in which service descriptions can be published and searched. Service consumers can find service descriptions from this directory and obtain binding information for the services therein.

Chapter 3

Iterative Bidding Model for Travel Service Customization

The use of auctions for allocating limited resources to competing customers is quite common and has been used for decades to assess different product and service prices on the market. With the advances of Internet technologies, web-based auctions have become an important way of linking providers' service capacities with end customers' needs. The proposed auction is a market-based mechanism in which an auctioneer offers a set of services to its customers and coordinates their customization requirements by adjusting the prices of service packages. In this chapter, we first describe the travel service customization problem model which uses customers' values as inputs and then finds the optimal solution. We subsequently introduce an iterative bidding procedure as the core mechanism of our web-based auction system for mass customization of travel services without requiring the valuations of customers.

3.1 The Travel Service Customization Problem

Being able to provide customers with the ability to individually design and customize products and services is very appreciable in today's market. It is observable that many travel companies have endeavoured to enhance the customization abilities of their travel services. However, a company's service capacities and a customers' values are tightly related. This section provides a description and formulation of the service customization problem model, wherein service customization under capacity constraint is examined as an optimization problem. This problem is formulated in a centralized sense, meaning that the service provider is assumed to haves access to all the required information to compute the optimal solution.

The travel service customization problem consists of a group of customers and a service provider. In this model the service provider offers a variety of services to customers in the form of a wide number of products. These products can then be customized by customers by selecting a pre-defined group of services and adding other optional services according to their preferences. The customized product is thus a package of various services selected by customers. The customized package includes a set of vacation services such as travel tickets services, accommodation services, and/or additional entertainment tickets. Each one of these services has a capacity limit that is provided by its service providers. When customers have finished defining all of their packages, it is the time for them to attach a value (the maximum price they are willing to pay) to each of their packages. Since this model follows the private value model of Vickery (1961), customers will attach their own value on each package privately such that this value does not depend on other customers' values. In this case, other individuals' values are not known to other customers. Accordingly, each customer will be willing to pay for their packages up to their own value, thereby maintaining positive payoffs. It is important to mention that the customers' value is fixed and is not the price that he or she may ultimately pay for a package. However, when the price of a customer's package increases, the related payoff decreases.

The service customization problem model consists of a set of n customers as well as a set of m services. Customers will customize their service package by selecting a set of services. As previously stated, a service package has to have a base configuration (a pre-configured set of services) which is denoted by $\bar{S}$. For Service i, its capacity is limited as $capacity(i)$. Let E_j be the set of service packages of customers and E be the union of the set of feasible packages from all customers $E = U_{j=1..n}E_j$. Let $v_j(B)$ be the value attached to the service package $B \epsilon E$

by customer j. In this case, $v_j(B) > 0$ if $B \epsilon E_j$ and $v_j(B) = 0$ otherwise. Let $x(B) = 1$ if the package $B \epsilon E$ is allocated to customer j; let $x(B) = 0$ otherwise.

This problem model selects the set of customer service packages in a way that the service providers' capacity constraints are respected while, at the same time, the sum of values of customers' selected packages are maximized. This model is formulated as the following integer programming:

$$\boldsymbol{max \sum_{j=1}^{n} \sum_{B \epsilon E} x_j(B) v_j(B)}$$

$\boldsymbol{subject\ to}$

$$\sum_{B \in E} \boldsymbol{x_j(B) \leq 1} \qquad j = 1, \dots, n \qquad (1)$$

$$\sum_{B \ni i} \sum_{j=1}^{n} \boldsymbol{x_j(B) \leq capacity(i)} \qquad i = 1 \dots m \qquad (2)$$

$$\sum_{B \in E} \boldsymbol{x_j(B)} = \sum_{B \in E_j} \boldsymbol{x_j(B)} \qquad j = 1, \dots, n \qquad (3)$$

$$\sum_{B \in E} \boldsymbol{x_j(B)} = \sum_{B \supseteq \overline{S}} \boldsymbol{x_j(B)} \qquad j = 1, \dots, n \qquad (4)$$

$$\boldsymbol{x_j(B)} = \{\boldsymbol{0}, \boldsymbol{1}\}, \quad \boldsymbol{B \in E} \qquad j = 1, \dots, n \qquad (5)$$

Constraint (1) makes sure that a customer can only obtain one travel package. Constraint (2) ensures that the allocation of customers' packages does not exceed the provider's capacity limit. The set of Constraint (3) ensures that if a package is assigned to a customer, this package must belong to a set of product configurations acceptable by customers. Constraint (4) serves to safeguard the base configuration in each awarded package. Constraint (5) is a set of integer constraints.

3.2 The Iterative Bidding Model

Our auction system is designed as an iterative combinatorial bidding procedure which is an alternative to simultaneous single-item auctions that would allow participants to submit multiple bids during an auction on packages, or combinations or bundle of items rather than a single item (Parkes & Ungar, 2000). This type of auction is usually used when customers' values are complementary or when under production and financial constraints (Porter, Rassenti, Roopnarine, & Smith, 2003). In this model, a customer's bid is represented as *(package, biddingprice),* where *package* is the set of travel services chosen by a customer and the *biddingprice* is the price that the customer is going to pay for that package. A reservation price is assigned to each package, so that customers start bidding on a package from the reservation price or higher. The reservation price is the minimum price for a service that the provider is willing to sell to its customers. In this situation, the service provider is able to sell a package, which contains a set of services, to its customers at different prices.

Bidding Procedure

The bidding procedure in our auction design consists of five main stages: (1) initialization, (2) updating price and submitting bid, (3) bid screening, (4) termination checking, and (5) winner determination. Figure 3-1 shows the steps of how these stages take place in the auction. Details of each step are described below.

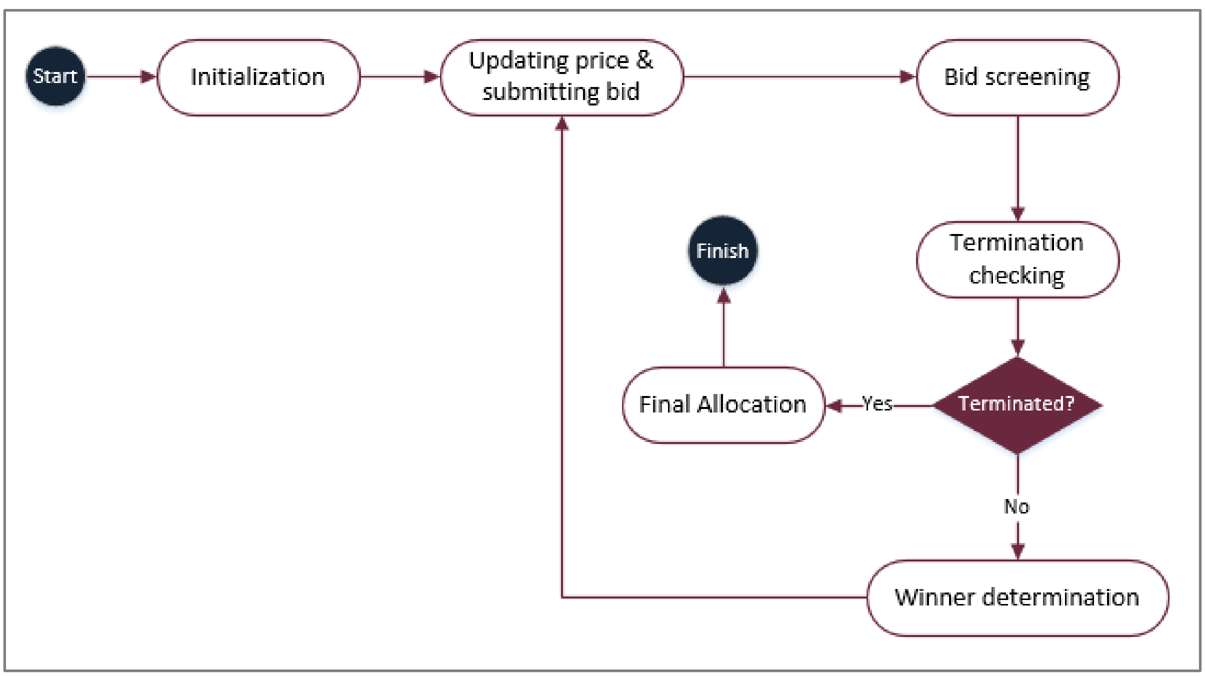

Figure 3-1: Iterative Bidding Procedure

3.2.1 Initialization

Before bidding starts, the service provider presents a set of available services to customers and identifies the services that must be included in each package as a base configuration. Customers will then select a set of services as their set of feasible packages E_j, compute a value, and attach the value to each package in E_j. Customers are allowed to customize a limited number of feasible packages for bidding. This limit is set to five per customer in our web-based auction system. Normally, the service provider assigns a reservation price for each package. As previously discussed, the reservation price is the minimum price of each package that the service provider can sell to its customers. The initial bidding price for each package is equal to its reservation price. At this point, the values and reservation prices are known, allowing customers to compute the payoff of each package. Payoff is the customers' value minus the initial bidding price. In order to keep positive payoff, customers will pay for their packages up to their chosen value for each package. Customers will then select the package that has the highest payoff as the first package on which to bid.

3.2.2 Updating price and submitting bid

In each round of auction t, a number customers will be awarded a package in the provisional allocation. At the beginning of each round where $t > 1$, customers will need to update the bidding prices of the packages that they submitted in round $t-1$, based on what they were awarded in the provisional allocation in round $t-1$. If a customer is included in the provisional allocation at round $t-1$ he or she can keep the bidding price unchanged in the next round. However, if a customer was not included in the provisional allocation, he or she will be given two price updating options:

1) The customer can increase the bidding price by ε for the package which he or she bid for in the previous round where ε is the minimum price increment set by the provider,
2) The customer can keep his or her bidding price unchanged. This means that the customer has taken a discount of ε. In this case, the provider considers that this customers has entered into *final bid status.* Accordingly, he or she will be no longer be allowed to increase the bidding prices of any of his or her packages in the future rounds.

After the prices are updated, customers select the package that has the highest payoff and submit it to the provider with the updated bidding price. If a customer has entered into the final bidding status he or she will not be permitted to increase his or her bidding price. However, customers can repeat the submission of the final bid during the rest of the auction's rounds. The aim of *this final bid repeating* is to boost the provider's revenue.

3.2.3 Bid screening

After the provider receives customers' bids, the provider first screens out any invalid bids. Submitted bids are considered as invalid when: (1) the base configuration is not included in a package, (2) the packages' prices increased from those of the customers who have already announced their final bidding status in previous rounds, and (3) the bidding price for a package that is lower than the bidding price that same package received in previous rounds.

3.2.4 Termination checking

The provider needs to check the termination condition in each round of the auction. The bidding procedure terminates if the provider does not receive any updated prices for the current round. This means that all the customers have repeated the same bid in the current round as in the previous round. When the bidding procedure terminates, the provider conducts the final allocation. However, if termination is not applied, the provider will take the set of valid bids and compute the winner determination model. After the winner determination is solved, the auction goes back to the updating price and submitting bid stage.

3.2.5 Winner determination

As long as the bidding process is not terminated, the winner determination problem must be solved and a provisional allocation must be obtained in each round. The winner determination model is designed so that a subset of the submitted bids will be selected, causing the overall bidding price of the provisional allocation to be maximized and preventing the provider's capacity constraints for each service from being violated. The winner determination problem is formulated as the following integer programming model:

$$max \sum_{j \in N^t} p\ (B_j^t)$$

$$subject\ to$$

$$\sum_{\substack{j \in N^t \\ B_j^t \ni i}} Z_j \leq capacity\ (i), \quad i = 1 \dots m \qquad (6)$$

$$Z_j = \{0,1\}, j \in N^t \qquad (7)$$

N^t is the set of customers who submitted bids at round t. B_j^t is the package that is submitted by customer j in round t, where $j \in N^t$ and $p\ (B_j^t)$ is the bidding price of that package. Let $Z_j = 1$ if customer j wins and $Z_j = 0$ otherwise. Constraint (6) ensures that the packages assigned in the provisional allocation do not breach the provider's capacity constraints. Constraint (7) is a set of integer constraints.

3.2.6 A worked example

Table 3-1: Customers' Travel Packages and Corresponding Initial Price and Value

Customer	Travel Package	Initial Price	Value
C#1	B(1,1)={DT3, RT3, HL3, ET2, ET3}	$2,470	$2,505
	B(1,2)={DT2, RT1, HL1, ET1, ET3, ET4, ET5}	$3,260	$3,310
	B(1,3)={DT3, RT2, HL2, ET4}	$2,890	$2,930
C#2	B(2,1)={DT1, RT3, HL1, ET2, ET1}	$3,180	$3,205
	B(2,2)={DT1, RT1, HL1, ET1, ET4, ET5}	$3,380	$3,420
	B(2,3)={DT3, RT3, HL2, ET1, ET3, ET4}	$3,120	$3,145
C#3	B(3,1)={DT3, RT3, HL1, ET1, ET2, ET3, ET5}	$3,270	$3,295
	B(3,2)={DT1, RT2, HL1, ET1, ET3, ET4}	$3,420	$3,430
C#4	B(4,1)={DT2, RT1, HL3, ET1, ET2, ET3, ET4}	$2,720	$2,755
	B(4,2)={DT2, RT1, HL2, ET1, ET3}	$2,790	$2,800
C#5	B(5,1)={DT3, RT3, HL2, ET5}	$2,670	$2,720
	B(5,2)={DT1, RT3, HL1, ET4, ET5, ET1, ET3}	$3,420	$3,430
	B(5,3)={DT2, RT2, HL1, ET4, ET5, ET2}	$3,190	$3,195
	B(5,4)={DT3, RT3, HL3, ET3, ET5}	$2,350	$2,355
	B(5,5)={DT1, RT3, HL2, ET3, ET4}	$2,970	$3,005

In this section we provide a worked example in order to demonstrate the application of the iterative bidding model. The example is based on the customization of travel packages. In this example the service provider provides customers with a list of travel components. These components are identified as Destination Ticket (DT), Return Ticket (RT), Hotel (HL), and Entertainment Ticket (ET). Each component has different services. For example DT has schedule in the morning (DT1), in the evening (DT2), or at night (DT3). There are multiple services for the other components such as RT, HL, and ET, each of which has a capacity. Table 3-1 shows customers' travel packages, the initial price and value of their packages where B(a, b) shows travel package b from customer a.

Table 3-2: Submitted Bids, Provisional Allocation, Provider's Revenue, and Customer's Value at each Round of Bidding

Round#	Submitted Bids & Individual Bidding Prices	Provisional Allocation & Individual Customers' Values	Sum of Provider Revenue	Sum of Customer Value
	$3,260 $3,380 $3,270 $2,720 $2,670	$3,420 $2,755 $2,720		
1	B(1,2), B(2,2), B(3,1), B(4,1), B(5,1)	B(2,2), B(4,1), B(5,1)	$8,770	$8,895
	$3,265 $3,380 $3,275 $2,720 $2,670	$3,420 $2,755 $2,720		
2	B(1,2), B(2,2), B(3,1), B(4,1), B(5,1)	B(2,2), B(4,1), B(5,1)	$8,770	$8,895
	$3,270 $3,380 $3,280 $2,720 $2,670	$3,420 $2,755 $2,720		
3	B(1,2), B(2,2), B(3,1), B(4,1), B(5,1)	B(2,2), B(4,1), B(5,1)	$8,770	$8,895
	$2,890 $3,380 $3,285 $2,720 $2,670	$2,930 $3,420 $2,720		
4	B(1,3), B(2,2), B(3,1), B(4,1), B(5,1)	B(1,3), B(2,2), B(5,1)	$8,940	$9,070
	$2,890 $3,380 $3,420 $2,725 $2,670	$2,930 $3,430 $2,720		
5	B(1,3), B(2,2), B(3,2), B(4,1), B(5,1)	B(1,3), B(3,2), B(5,1)	$8,980	$9,080
	$2,890 $3,385 $3,420 $2,730 $2,670	$2,930 $3,430 $2,720		
6	B(1,3), B(2,2), B(3,2), B(4,1), B(5,1)	B(1,3), B(3,2), B(5,1)	$8,980	$9,080
	$2,890 $3,390 $3,420 $2,735 $2,670	$2,930 $3,430 $2,720		
7	B(1,3), B(2,2), B(3,2), B(4,1), B(5,1)	B(1,3), B(3,2), B(5,1)	$8,980	$9,080
	$2,890 $3,180 $3,420 $2,740 $2,670	$2,930 $3,205 $2,755 $2,720		
8	B(1,3), B(2,1), B(3,2), B(4,1), B(5,1)	B(1,3), B(2,1), B(4,1), B(5,1)	$11,480	$11,610
	$2,890 $3,180 $3,290 $2,740 $2,670	$2,930 $3,205 $2,755 $2,720		
9	B(1,3), B(2,1), B(3,1), B(4,1), B(5,1)	B(1,3), B(2,1), B(4,1), B(5,1)	$11,480	$11,610
	$2,890 $3,180 $3,425 $2,740 $2,670	$2,930 $3,205 $2,755 $2,720		
10	B(1,3), B(2,1), B(3,2), B(4,1), B(5,1)	B(1,3), B(2,1), B(4,1), B(5,1)	$11,480	$11,610
	$2,890 $3,180 $3,295 $2,740 $2,670	$2,930 $3,205 $2,755 $2,720		
11	B(1,3), B(2,1), B(3,1), B(4,1), B(5,1)	B(1,3), B(2,1), B(4,1), B(5,1)	$11,480	$11,610
	$2,890 $3,180 $3,295 $2,740 $2,670	$2,930 $3,205 $2,755 $2,720		
12	B(1,3), B(2,1), B(3,1), B(4,1), B(5,1)	B(1,3), B(2,1), B(4,1), B(5,1)	$11,480	$11,610

The initial price (reservation price) is the minimum price that a customer should pay for a package. The value of each customer's package is generated using the method described in the "Design of the Testing Data" section. In this example each package has to have one DT, one RT, and one HL as the base configuration. Customers can select from one to five ET components. To reduce the number of iterations, high reservation prices are provided for each package (see Table 3-1). Table 3-2 summarizes the number of rounds, submitted bids, provisional allocation, provider revenue, and customer value of each round. Additionally, individual bidding prices and customers' value are provided above each round of auction. For the auction process the fixed price increment ε has been set at 5. As it can be seen, the auction terminates at round 12 with a total customer value at \$11,610. Compared with the optimal value \$12,030, the auction reaches 96% efficiency in this example. In addition, the obtained provider revenue is \$11,480 which is close to the overall customer value. Since the customer value is hidden from the service provider in the iterative bidding model, our web-based auction system as the Auctioneer will act as the intermediary between customers and service providers in order to hide the value of customers and submit bids on behalf of them. It is also important to mention that, customers' values in this example are generated randomly for testing the system and in real case scenarios they should be hidden from the service provider.

Chapter 4

System Requirements Analysis

In this chapter we define the requirements of our system in detail. We start by explaining the main goal of our web-based auction system as a whole: to enable customers to submit bids on available travel services such that services' capacity constraints are respected and the sum of customer value taken from the selected packages is maximized. However, there are other important requirements that must be carefully considered in order to satisfy this goal. In the following sections we define our system requirements from three different perspectives: those of the Bidders (customers), the Manager (system controller), and the Auctioneer (web application).

4.1 Bidder and Manager

The Bidder acts as an actor to our system. The system is designed in a way that it provides Bidders with the following functional scenarios:

Register. Bidders must register into the system. This is needed in order to identify the submission of bids in the auction system. The auction system further requires a Bidder's information to assign them packages.

Add package to cart. Bidders will be able to submit bids on their travel packages. To facilitate this, it is necessary to provide them with an environment in which they can easily select services and add them to their shopping cart as packages.

Delete package. Of course Bidders may change their mind about a package over the course of their browsing. Consequently they should be able to easily delete one or all of their packages.

Delete cart. This function will provide Bidders with the ability to delete their entire shopping cart at once in the case that they plan not to submit any bid at all.

Submit bid. After readying the packages in their shopping carts, Bidders will have the options to submit a bid on either one or all of their travel packages at the same time.

Since the auction system needs to be controlled by someone with grant access to different parts of the system, we assigned an actor called Manager to control the system from requesting travel services until it has terminated an auction. The Manager will be needed to have the following responsibilities and interactions with our web-based auction system:

Request travel services. In order for an auction to begin, there should be travel services stored in the system's database. This is needed so that Bidders will be able to view and submit bids on those services. For this reason, the Manager needs to send a request for travel services to travel web services. Upon receiving the services, our Manager will have the option to store services.

Start auction. Each auction should be initiated by the Manager. The Manager will be able to do so from a Web browser which is designed for him or her. When an auction is started, the Bidders will be able to view the services.

Terminate auction. The Manager is responsible for terminating an auction when the auction should be terminated. The termination time depends on the duration and the validity of travel services.

View auction results. When an auction is terminated by the Manager and the results of that auction are available the Manager will be able to view the results.

View current services. At any time the Manager should be able to view the current services that are already requested from travel web services and are stored in the database by the Manager. He or she will then be able to check each service's details. Figure 4-1 is a UML Use Case diagram that illustrates how Bidders and the Manager interact with our auction system.

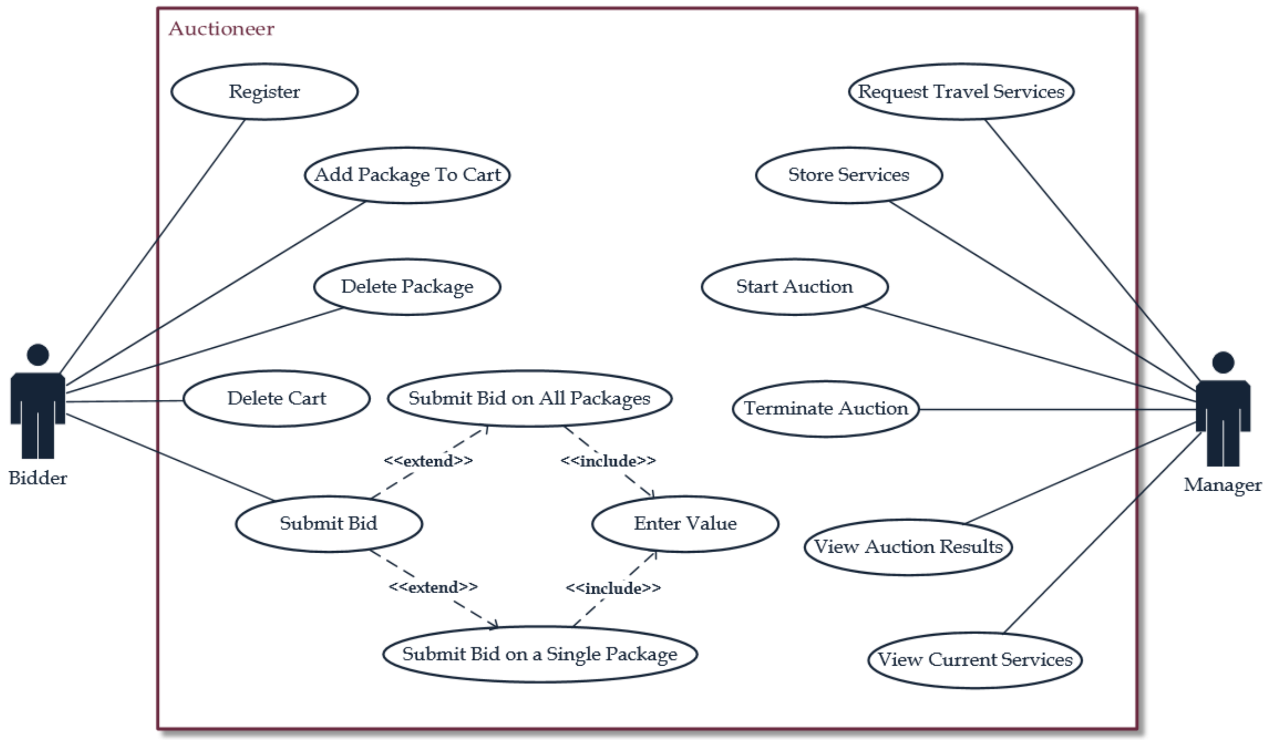

Figure 4-1: UML Use Case Diagram for Bidder and Manager

In order to illustrate the workflow of the Bidder scenarios, we have depicted the following UML Activity diagram. Figure 4-2 shows a range of possible situations that may possibly be encountered by a Bidder in our web-based system. These begin with a registration or log in. By doing so, the Bidder will be able to see the available travel services which are provided by the Auctioneer, and act accordingly.

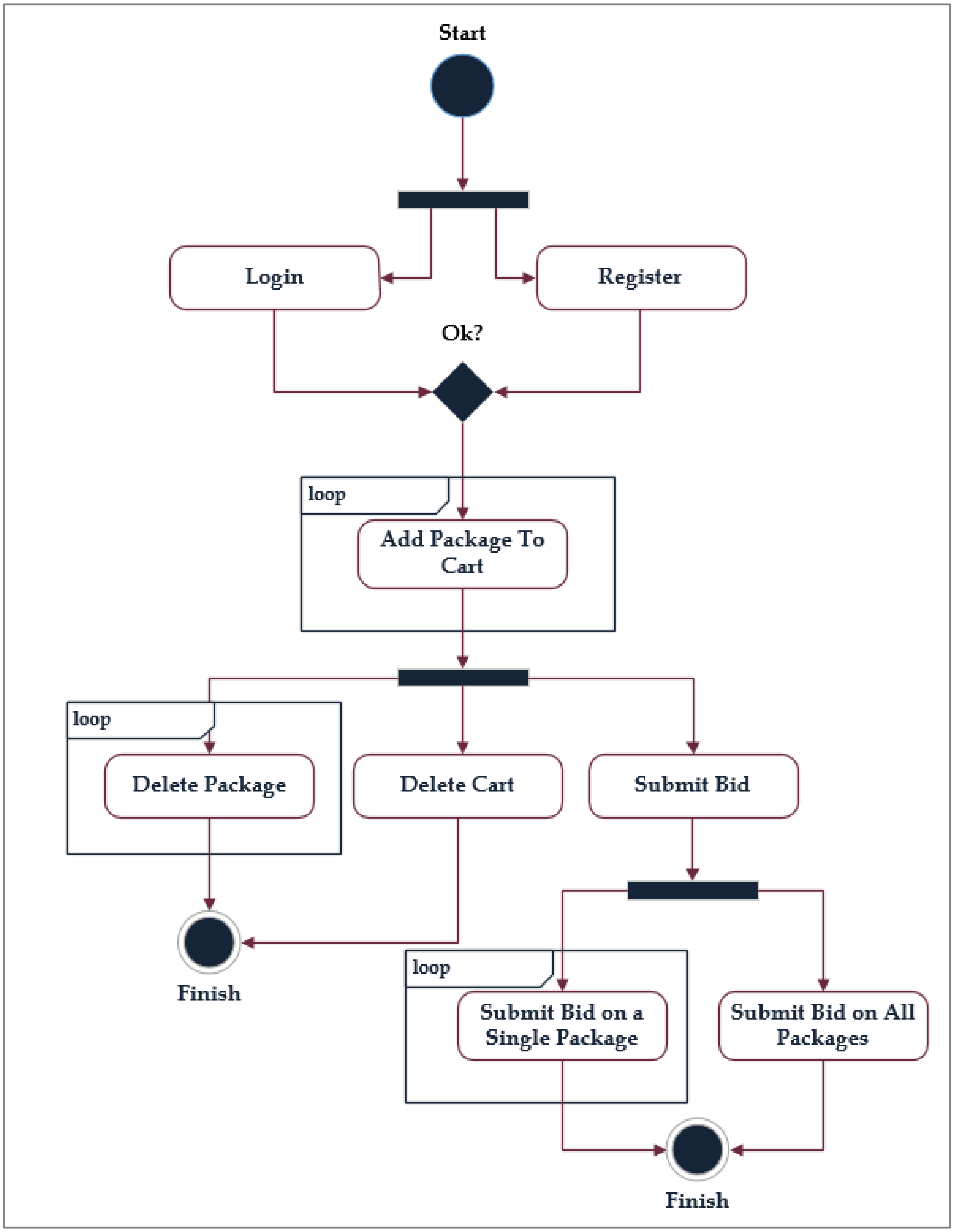

Figure 4-2: UML Activity Diagram Showing Workflow of Bidders' Scenarios

At this point, Bidders are provided with the ability to select, customize, and add services to their shopping carts as different packages. Now they have the option to delete a package, delete the cart, or submit bids on their packages. In order to submit bids, a Bidder must choose between two available options: to either submit a bid on a single package or to submit a bid on all packages together. By submitting bids, the Auctioneer will store each Bidder's packages into its database so that when the auction begins these packages can be obtained from the database. Bidders should then wait to receive the auction results. The detailed descriptions of each scenario for the actor "Bidder" are provided as follow:

Table 4-1: Login Use Case Description for Bidder

Use Case:	**Login**	
Brief Description:	Bidder wants to login into the system.	
Actor:	Bidder	
Preconditions:	A Bidder must be registered into the system.	
Postconditions:	User must be logged in.	
Main flow of events:	**Actor**	**System**
	1. Bidder clicks on "Login" link.	1.1. System opens the "Login" section.
	2. Bidder fills in required details.	
	3. Bidder clicks on "login" button.	3.1. System logs Bidder in.
Exception Conditions:	3.1. If the Bidder does not enters the required fields correctly, the system will ask him or her to try again.	

Table 4-2: Register Use Case Description for Bidder

Use Case:	**Register**	
Brief Description:	Bidder wants to register into the system and create a new account.	
Actor:	Bidder	
Preconditions:	A Bidder must have access to the system.	
Postconditions:	An account must be created for the Bidder.	
Main flow of events:	**Actor**	**System**
	1. Bidder clicks on "Register" link.	1.1. System opens the "Register" section.
	2. Bidder fill in required details.	
	3. Bidder clicks on "Register" button.	3.1. System creates a new account. 3.2. System sends activation email.
Exception Conditions:	3.1. If the Bidder does not enter all the required fields, the system will not create the account and he or she needs to register again.	

Table 4-3: Delete Package Use Case Description for Bidder

Use Case:	**Delete Package**	
Brief Description:	Bidder wants to delete a package.	
Actor:	Bidder	
Preconditions:	1. Bidder must be logged in to the system. 2. Bidder must have a package in his or her cart.	
Postconditions:	The package must be deleted from the cart.	
Main flow of events:	**Actor**	**System**
	1. Bidder clicks on "Delete" button.	1.1 System deletes the package data from database. 2.1 System deletes the package from the cart's session.
Exception Conditions:	If the package could not be deleted from the database or cart, the Bidder will be informed to try again.	

Table 4-4: Add Package to Cart Use Case Description for Bidder

Use Case:	Add Package to Cart	
Brief Description:	Bidder want to add his or her package to the cart	
Actor:	Bidder	
Preconditions:	1. A Bidder must be logged in to the system 2. Bidder must have selected and customized a package.	
Postconditions:	Package must be added to the cart.	
Main flow of events:	**Actor**	**System**
	1. Bidder selects services from the menu.	
	2. Bidder clicks on "Add to Cart".	2.1 System stores Bidder's package to database. 3.1 System will add the package to the cart's session.
Exception Conditions:	2.1 If Bidder has reached his or her limit on selecting packages, the system will not add the package to the cart and will not store the package to database.	

Table 4-5: Delete Cart Use Case Description for Bidder

Use Case:	Delete Cart	
Brief Description:	Bidder wants to delete his or her shopping cart.	
Actor:	Bidder	
Preconditions:	1. Bidder must be logged in to the system. 2. Bidder must have at least one package in his or her cart.	
Postconditions:	Bidder's cart must be deleted.	
Main flow of events:	**Actor**	**System**
	1. Bidder clicks on "Delete cart" button.	1.1 System deletes the Bidder's cart info from database. 1.2 System removes the Bidder's cart session.
Exception Conditions:	If the cart data could not be deleted from the database or be removed from the cart session, the Bidder will be informed to try again.	

Table 4-6: Submit Bid Use Case Description for Bidder

Use Case:	**Submit Bid**	
Brief Description:	Bidder wants to submit a bid on his or her package or packages.	
Actor:	Bidder	
Preconditions:	1. Bidder must be logged in to the system. 2. Bidder must have at least one package in his or her cart. 3. Bidder must have entered his or her maximum value for his or her package or packages.	
Postconditions:	A confirmation must be sent to Bidder.	
Main flow of events:	**Actor**	**System**
	1. Bidder clicks on Submit Bid.	1.1 System stores data to database. 1.2 System sends confirmation to Bidder.
Exception Conditions:	1.1 If the data could not be stored in the database due to technical difficulties, the Bidder will be informed.	

In order to illustrate the workflow of the interactions of a Manager with the rest of the system, we have provided the following UML Sequence diagram. Figure 4-3 shows the flow of the Manager's activities, from starting an auction and proceeding until viewing the auction results. As can be seen from this diagram, the Manager first needs to request available travel services from web services. Upon receiving these services, the Manager will store them into the database. Subsequently, the Manager can either view the currents services or start an auction. After the auction proceeds, the Manager will terminate the auction and will be able to view the results.

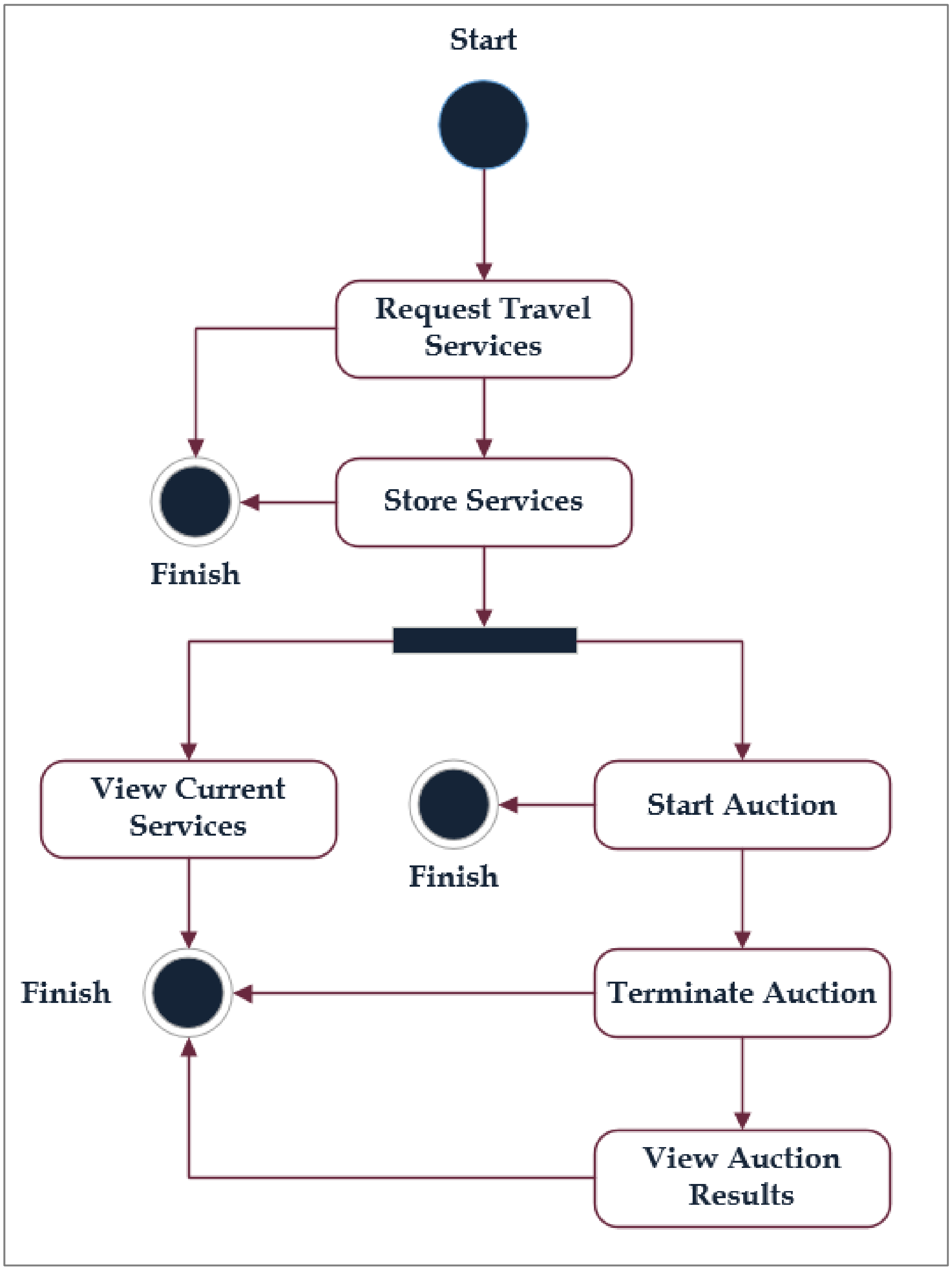

Figure 4-3: UML Activity Diagram Showing Workflow of Manager's Scenarios

For more detailed information of each use case, the following descriptions are documented:

Table 4-7: Request Travel Services Use Case Description for Manager

Use Case:	**Request Travel Services**	
Brief Description:	Manager wants to make a request for available travel services from travel web services.	
Actor:	Manager	
Preconditions:	1. Manager must be logged in to the system.	
Postconditions:	Travel services must be viewed by Manager.	
Main flow of events:	**Actor**	**System**
	1. Manager clicks on "Get Services".	1.1 System invokes flights web service. 1.2 System invokes hotel web service. 1.3 System invokes entertainment ticket web service. 1.4 System list services for Manager

Table 4-8: View Auction Results Use Case Description for Manager

Use Case:	**View Auction Results**	
Brief Description:	Manager wants to view an auction results.	
Actor:	Manager	
Preconditions:	1. Manager must be logged in to the system. 2. An auction must be previously processed. 3. Results must be stored in the database.	
Main flow of events:	**Actor**	**System**
	1. Manager click on "View Results".	1.1 System loads the auction results.

Table 4-9: Store Service Use Case Description for Manager

Use Case:	Store Services	
Brief Description:	Manager wants to store services to the database which are requested and received by travel web services.	
Actor:	Manager	
Preconditions:	1. Manager must be logged in to the system. 2. Manager must have requested travel services.	
Postconditions:	Travel services must be stored into the database.	
Main flow of events:	**Actor**	**System**
	1. Manager clicks on "Store Services".	1.1 System stores services into the database. 1.2 System sends confirmation to Manager.

Table 4-10: Start Auction Use Case Description for Manager

Use Case:	Start Auction	
Brief Description:	Manager wants to start a new auction.	
Actor:	Manager	
Preconditions:	1. Manager must be logged in to the system. 2. Services must have been requested by the Manager and be stored into the database.	
Postconditions:	The auction should be started. Bidders must be able to view this auction's services.	
Main flow of events:	**Actor**	**System**
	1. Manager clicks on "Start".	1.1 System updates services table from the database and activates services to be viewed by Bidders. 1.2 System sends confirmation to Manager.

Table 4-11: Terminate Auction Use Case Description for Manager

Use Case:	Terminate Auction	
Brief Description:	Manager wants to terminate an auction.	
Actor:	Manager	
Preconditions:	1. Manager must be logged in to the system. 2. An auction must be previously started.	
Postconditions:	1. Auction must be terminated. 2. Auction must be processed. 3. Confirmation must be sent to the Manager.	
Main flow of events:	**Actor**	**System**
	1. Manager clicks on "Terminate".	1.1 System deactivates services of that auction. Bidders will not be able view these services. 1.2 System will process the auction and computes the winner determination. 1.3 System stores results into database. 1.4 System sends confirmation to the Manager.

Table 4-12: View Current Services Use Case Description for Manager

Use Case:	View Current Services	
Brief Description:	Manager wants to view current services.	
Actor:	Manager	
Preconditions:	1. Manager must be logged in to the system. 2. Services must be requested from travel web services. 3. Services must be stored into the database.	
Main flow of events:	**Actor**	**System**
	1. Manager clicks on "Show Results".	1.1 System retrieves services from database. 1.2 System lists services to the Manager.

4.2 Auctioneer

In order for Bidders to be able to view travel services or submit bids, and in order for the Manager to control and manage auctions, we have designed a web application called the Auctioneer. The Auctioneer acts as an actor and interacts with travel web services. The Auctioneer has the following functional activities:

Get flight services. Upon the Manager's request, the Auctioneer will invoke the Flight Web Service in order to receive the available flight services.

Get hotel services. The Auctioneer will also receive hotel services by the Manager's request by invoking the Hotel Web Service through our web application.

Get entertainment ticket services. Entertainment tickets such as sporting tickets, museum tickets, art tickets etc. will be received by the Auctioneer from the Entertainment Ticket Web Service upon the Manager's request.

These functionalities have been illustrated in the following use case diagram:

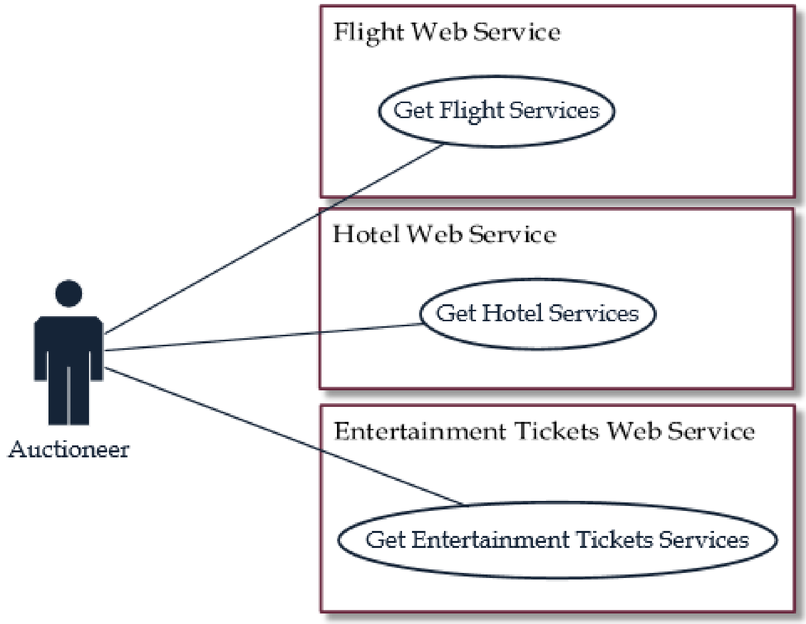

Figure 4-4: UML Use Case Diagram for Auctioneer

The detailed descriptions of each use case for the actor Auctioneer are provided below:

Table 4-13: Get Flight Services Use Case Description for Auctioneer

Use Case:	**Get Flight Services**	
Brief Description:	Auctioneer wants to get flight services from the flight web service.	
Actor:	Auctioneer	
Preconditions:	Manager must have clicked on the "Request Services" button.	
Postconditions:	Flight services must be sent to the Auctioneer.	
Main flow of events:	**Actor**	**System**
	1. Auctioneer invokes the flights web service to get available flights. 2. Auctioneer returns the list to the Manager.	1.1 Flight web service will search for flights. 1.2 Flight web service returns a flight list.

Table 4-14: Get Hotel Services Use Case for Auctioneer

Use Case:	**Get Hotel Services**	
Brief Description:	Auctioneer wants to get hotel services from the hotel web service.	
Actor:	Auctioneer	
Preconditions:	Manager must have clicked on the "Request Services" button.	
Postconditions:	Hotel services must be sent to the Auctioneer.	
Main flow of events:	**Actor**	**System**
	1. Auctioneer invokes the hotel web service to get available hotels. 2. Auctioneer returns the list to the Manager.	1.1 Hotel web service will search for hotels. 1.2 Hotel web service returns a hotel list.

Table 4-15: Get Entertainment Ticket Services Use Case Description for Auctioneer

Use Case:	**Get Entertainment Ticket Services**	
Brief Description:	Auctioneer wants to get entertainment ticket services from the hotel web service.	
Actor:	Auctioneer	
Preconditions:	Manager must have clicked on the "Request Services" button.	
Postconditions:	Ticket services must be sent to the Auctioneer.	
Main flow of events:	**Actor**	**System**
	1. Auctioneer invokes the entertainment ticket web service to get available tickets. 2. Auctioneer returns the list to the Manager.	1.1 Entertainment ticket web service will search for entertainment tickets. 1.2 Entertainment ticket web service returns a ticket list.

Chapter 5

System Design and Implementation

In this chapter we describe how our system is designed based on the requirements that we defined in our Requirement Analysis chapter. We start by illustrating the architectural design of our auction system as a whole and then we describe how our system components can communicate with each other. Finally, the major components of our auction system will be described in more detail.

5.1 System Architecture

Figure 5-1 illustrates a high level architectural view of our auction system as a whole. As it illustrates, our auction system is composed of five main parts: User, Auctioneer, Flight Web Service, Hotel Web Service, and Entertainment Ticket Web Service. As described previously, Users include both the Bidders and the Manager who use and interact with the Auctioneer. The Auctioneer is our web application which provides different types of access through different user interfaces for both the Manager and the Bidders.

The Auctioneer has interactions with web services. We have designed these 3 web services, so that the Manager can request travel services such as hotels, flights, and entertainment tickets dynamically at the beginning of an auction. The interactions of each actor are described in the Requirement Analysis chapter.

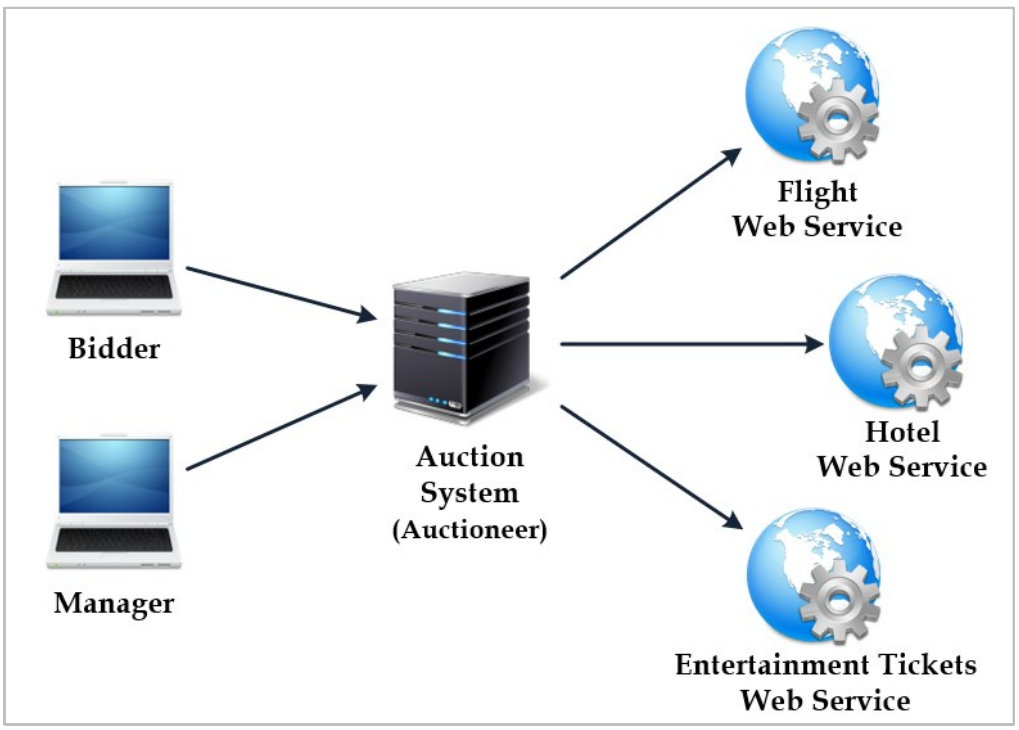

Figure 5-1: Architectural Diagram for the Web-Based Auction System

In this section we provide a lower level view of our system architecture using UML Component diagram. We describe the main components of our system, their interactions and interfaces. Figure 5-2 shows how the components of our system interact with each other. As it can be seen from the diagram, the Auctioneer component contains three major sub-components: Service Control, Auction Control, and Winner Determination. Web Service components provide the travel services that will be consumed by the Auctioneer. Each one of these components contains a set of collaborating classes.

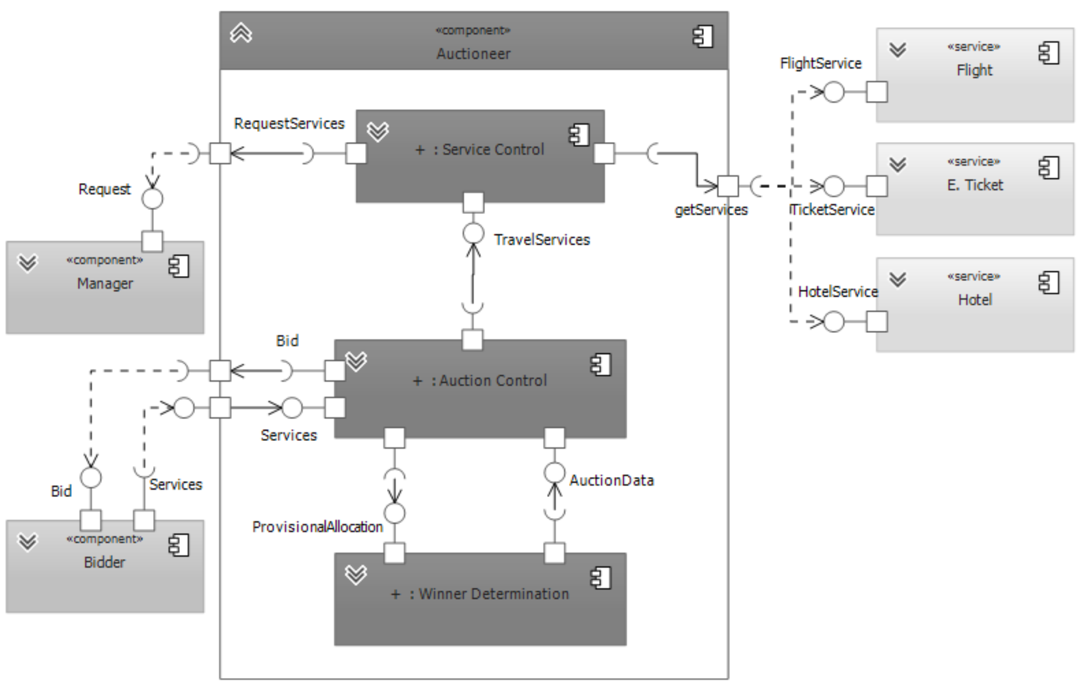

Figure 5-2: UML Component Diagram for the Web-Based Auction System

The Service Control component is responsible for controlling service issues in the auction system. For example, upon the Manager's request it will invoke each web service one by one. Each web service is provided with an interface so that the Auctioneer can invoke them. The Auction Control on the other hand, provides travel services to Bidders through a web browser so that Bidders can view those services and submit bids on them. In order for the Winner Determination component to function and compute the winners of each round, it requires the Auction Control to receive auction data in order to compute and provide the provisional allocation of each round. The Bidder component provides the bid interface, though it needs services and a required interface in order to be able to submit bids. The rest of this chapter will describe our main auction systems' components such as the web services and the Winner Determination components in more detail. However, due to the complexity of the system, separate class diagrams implemented using Java and their descriptions are given for each component.

5.1.1 Travel Web Services

As previously described, our auction system contains three different Web Services that are provided to serve the Auctioneer with multiple travels services.

Each of these web service has its own classes and attributes. The design of our Web Services are described as follows:

Flight Web Service

We have designed a Flight Web service in order to provide the Auctioneer with different flight schedules so that the Auctioneer can provide Bidders with these flight services. The UML class diagram in the following figure shows the classes, attributes, and methods of our flight service.

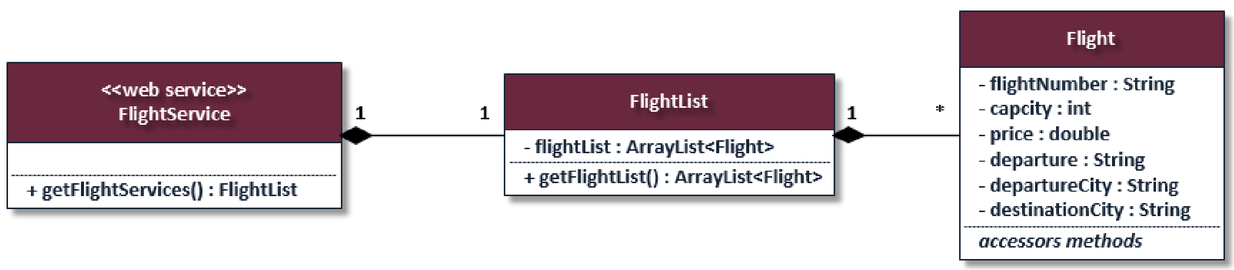

Figure 5-3: UML Class Diagram for Flight Web Service

As this diagram illustrates, the FlightService web service class is composed of one FlightList which is further composed of many Flight objects. Consequently, the FlightService class can be given as a provided interface for the Service Control component which needs to be invoked. Upon the invocation of this web service the Auctioneer will receive a list containing many objects of the Flight class. All the services that the Auctioneer receives from the Flight web service are searched through an XML file. The elements of this XML file will be obtained and will be stored in Flight objects to be sent to the Auctioneer.

Hotel Web Service

Our Hotel web service is designed in a way that it can be invoked by the Service Control component in order to provide the available hotel services. Figure 5-4 shows the class diagram of the Hotel web service. As it can be seen, the Hotel web service is composed of a HotelList that has as ArrayList attribute which contains many objects of Hotel. The Hotel web service also contains an XML file, so that multiple hotel descriptions can be stored in it.

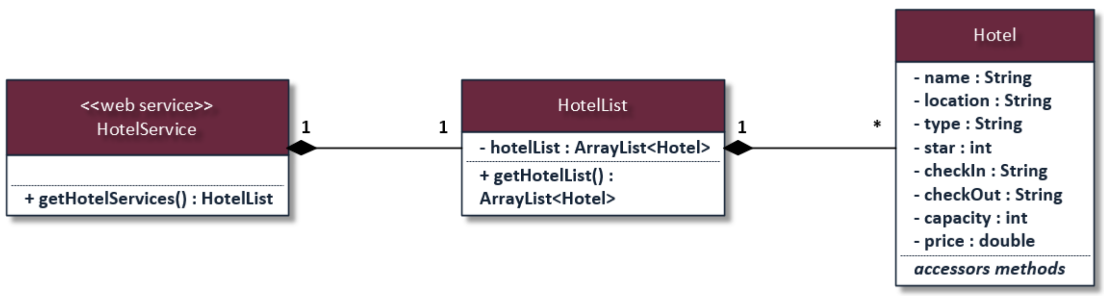

Figure 5-4: UML Class Diagram for Hotel Web Service

Entertainment Ticket Web Service

Like the Hotel and Flight web services, the Entertainment Ticket web service is designed to serve the Auctioneer with a set of various ticket services. There are various types of entertainment tickets provided with this web services such as art tickets, museum tickets, sporting tickets and so on. The following figure shows the class diagram of the Entertainment Ticket web service with its attributes and methods.

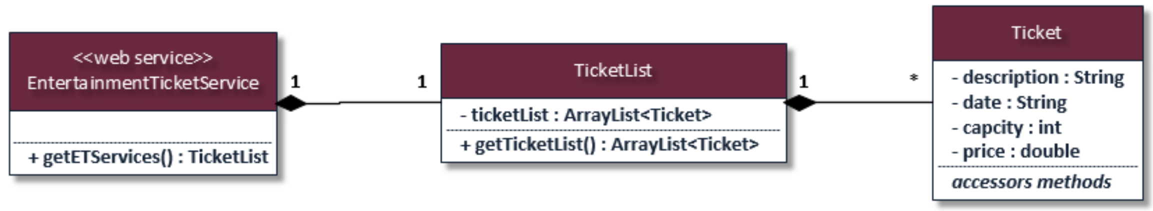

Figure 5-5: UML Class Diagram for Entertainment Ticket Web Service

This class diagram shows that the Entertainment Ticket web service is composed of one TicketList class which is itself composed of many Ticket objects. As in the case of the other web services, this web service contains an XML file for storing ticket information.

5.1.2 Service Invocation

Now that we have illustrated and described our three web services, we can explain how the Manager requests services and how the Auctioneer component invokes Java travel web services. As it is depicted in Figure 5-2, the Service Control component requires the web services' interfaces (WSDL) in order to be able to invoke them as it provides the services as an interface to the Auction Control component. In the following diagram we have provided a class diagram in order to show the classes which belong to the Service Control component and how these classes interact with the web services' classes. As it can be seen, the classes that belong to this component are ServiceSAO, which is responsible for invoking travel services, and the ServiceServlet, which receives the Manager's request for receiving travel services. These two classes create objects from ServiceList and ServiceBean classes so that travel services can be added to them.

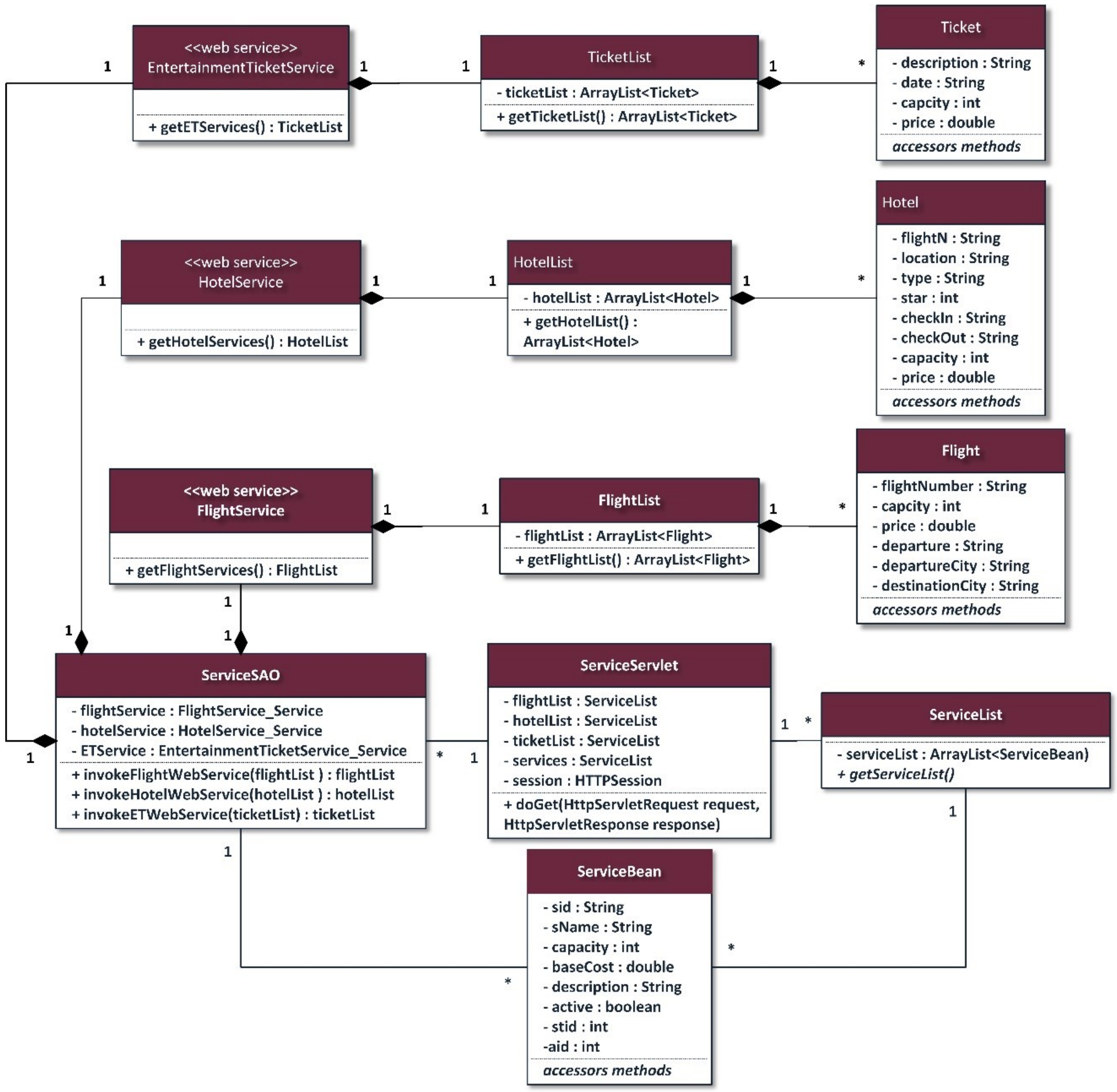

Figure 5-6: UML Class Diagram Showing Classes for Web Services' Invocation

In order to better understand the sequence of steps of how the Manager request travel services and how the Service Control invokes travel web services we have provided the following UML Sequence diagram:

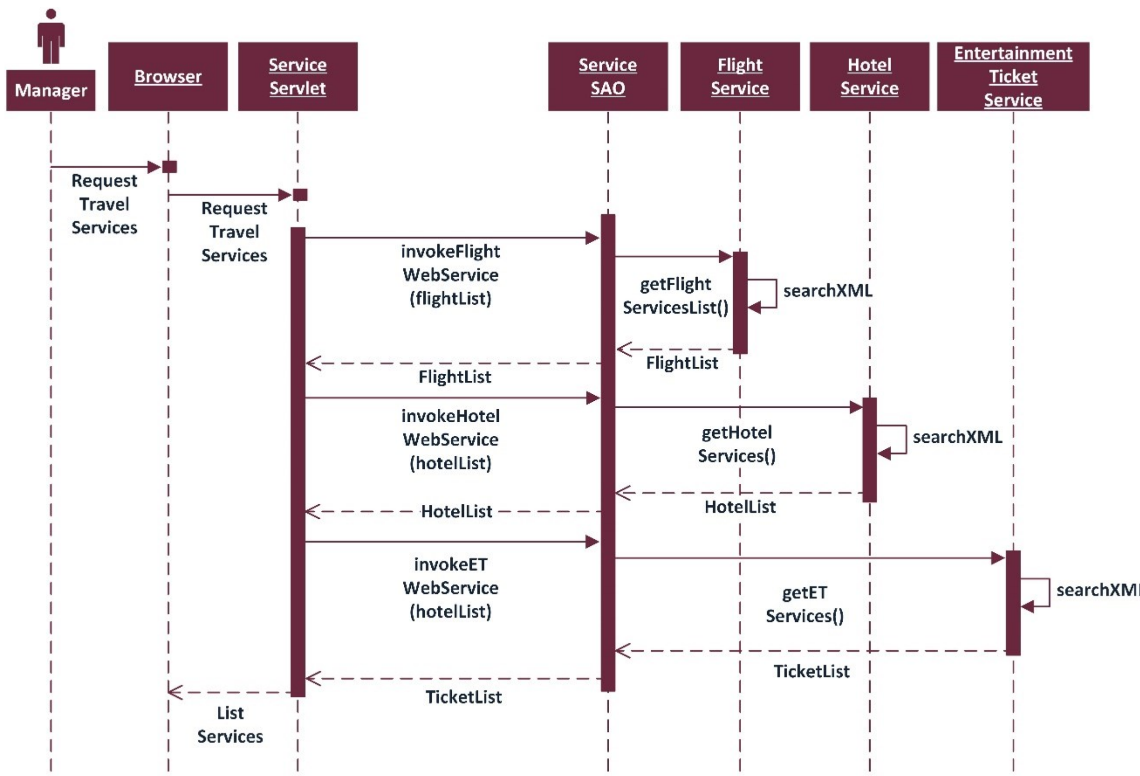

Figure 5-7: UML Sequence Diagram Showing Steps of Invoking Travel Web Services

As this sequence diagram illustrates, upon the Manager's request through the browser the ServiceServlet gets and sends the Manager's request to ServiceSAO. The ServiceSAO, which is a Java Servlet class, will then invokes the FlightService. The FlightService subsequently will search through its XML file in which all the flight information is stored, returning flight details as a FlightList object to the ServiceSAO. Each time the ServiceServlet receives travel objects it stores them in session beans. This process will continue in the same manner for the HotelService and the EntertainmentTicketService. Finally, when the ServiceServlet receives all the services and stores them in session beans, it returns them to the browser so that the Manager will be able to view them and choose whether or not to store them into database.

At this point, the Manager has requested travel services and the Service Control has consumed the requested travel services and stored them into the database. After the Manager starts an auction the Bidders will be able to view the available travel services so that they can select and customize them into multiple packages. To do so, the Auctioneer provides its services through the web browser. After registering or logging in, Bidders will be able to submit bids. Customer details and their submitted travel packages will be stored into the database. When the auction's end time arrives, the Manager needs to terminate the auction and the Auctioneer must run the auction process and solve the winner determination model. The Winner Determination (WD), which is described briefly at the beginning of this chapter, is a set of classes that work together in order to solve the problem model described in section 3.2.5 in order to find the winners of each round of an auction. For the WD to work and function there must be other classes which need to prepare and gather the required data to feed the WD module for processing an auction. In the following section we start by describing the WD and then explain how data is prepared for feeding the WD module.

5.1.3 Winner Determination

As previously described in section 3.2, the winner determination model computes a new provisional allocation in each round of an auction before the bidding termination. Figure 5-8 shows the Input Process Output (IPO) model of our winner determination procedure during each round of an auction. In order for the WD to function as a Java class, it needs to receive auction data as input. Auction data includes Bidders' information, their packages, and the services which belong to that auction. Upon receiving this data as an input, the WD transforms the data to an OPL data source. OPL is an Optimization Programming Language provided by IBM for solving combinatorial customization problems (IBM, 2009). The OPL data source is a java class which transforms Java objects to OPL objects.

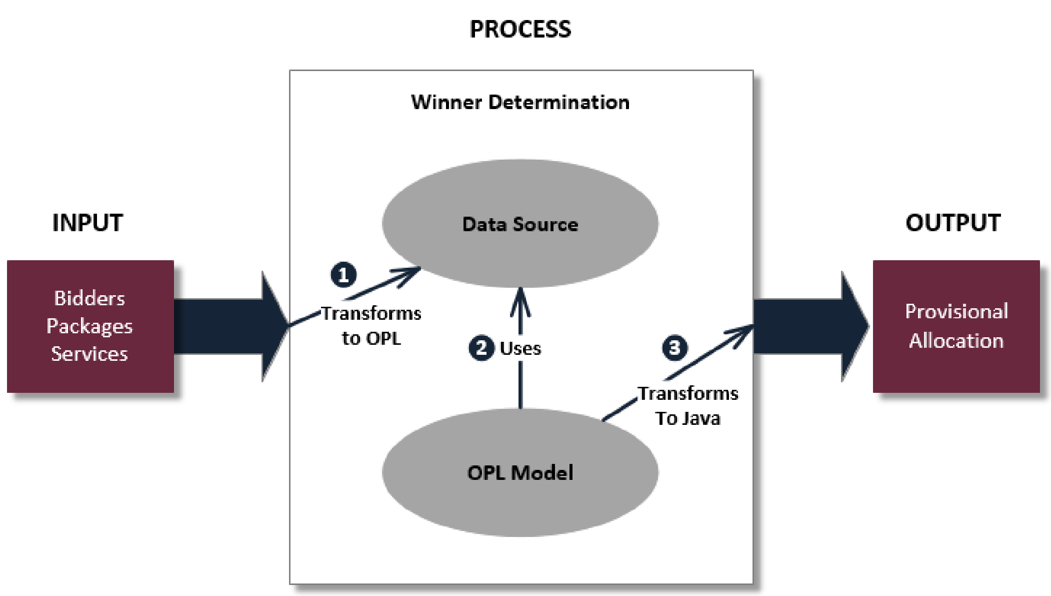

Figure 5-8: Winner Determination's IPO model

Following these actions, the OPL model in which our problem model is written (see section 3.2.5) will use the OPL data source as input to solve the winner determination model. Finally, the results will be obtained from the OPL model, transformed into Java objects, and returned as the process output. This output is called the Provisional Allocation and will be used for the next round of the auction. The whole process, from the starting of an auction to its end, is all about objects. For example, in the input of our model there are Bidders, packages and services whereas in the output there is a provisional allocation. These names are all objects. The following class diagrams model each of these objects in detail.

Figure 5-9 shows the objects that the Winner Determination class receives as input. The WD will transforms the BiderList, PackageList, and ServicesList objects into OPL as the data source so that the OPL Model can use this data to solve the problem model.

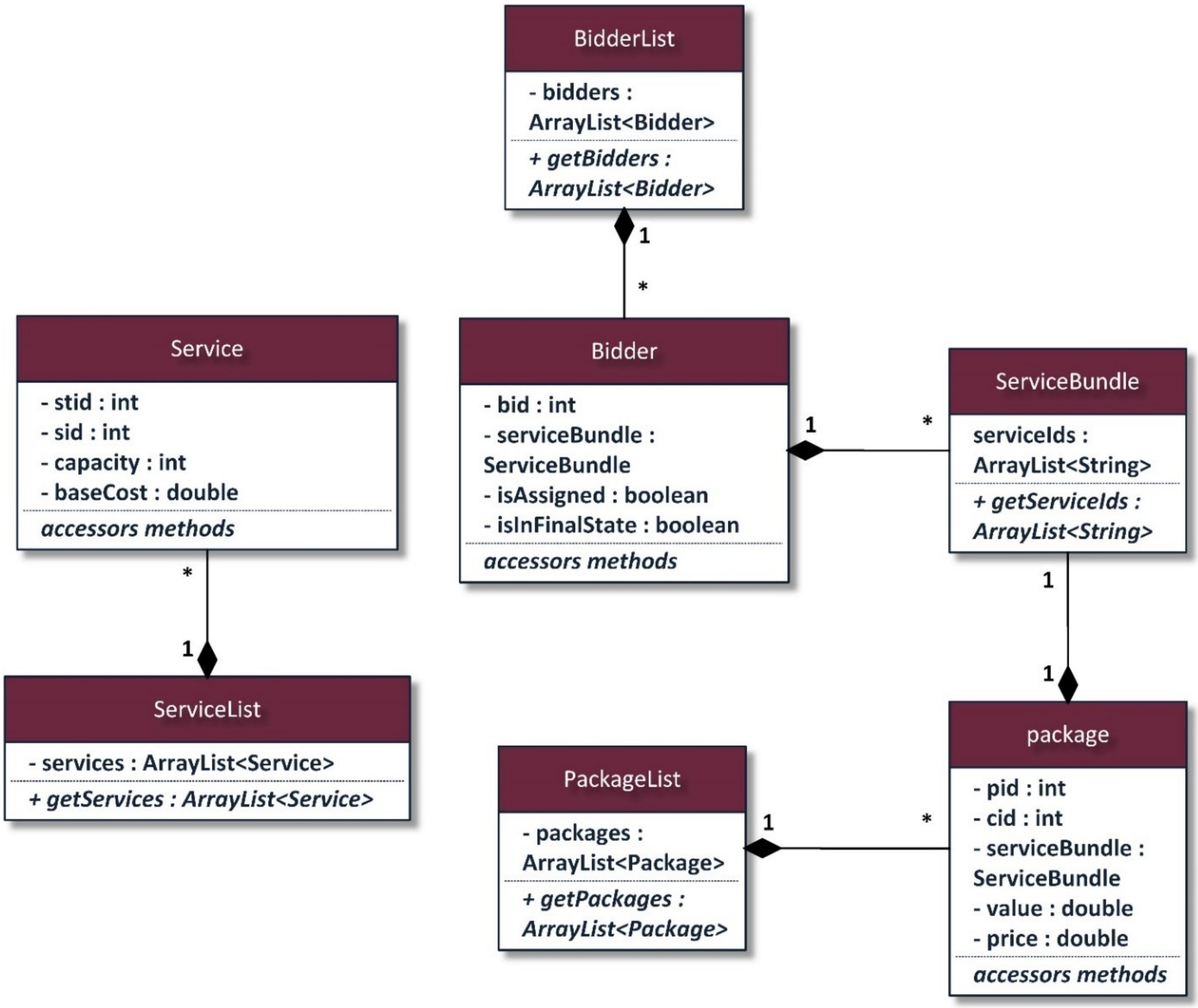

Figure 5-9: UML Class Diagram Showing the Objects of the Winner Determination Model

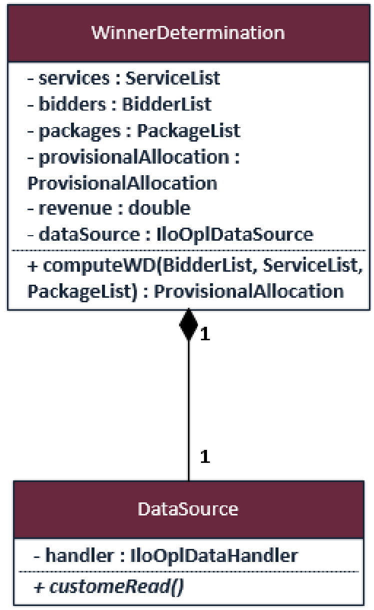

Figure 5-10: UML Class Diagram for Winner Determination

Figure 5-10 shows the winner determination objects as well as the data source class that it is composed of. As illustrated above, there is a method called computeWD which receives Bidders, packages, and services as parameters (input) and returns the ProvisionalAllocation as output. The ProvisionalAllocation contains packages allocated to Bidders in each round of auction. Classes which belong to the ProvisionalAllocation are depicted in Figure 5-11. Each ProvisionalAllocation is composed of a PackageList which is in turn composed of many Packages each containing a ServiceBundle in which service IDs are stored.

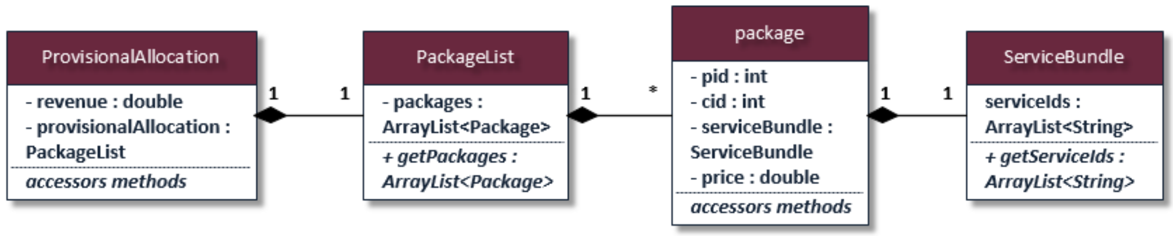

Figure 5-11: UML Class Diagram Showing the Provisional Allocations' Objects

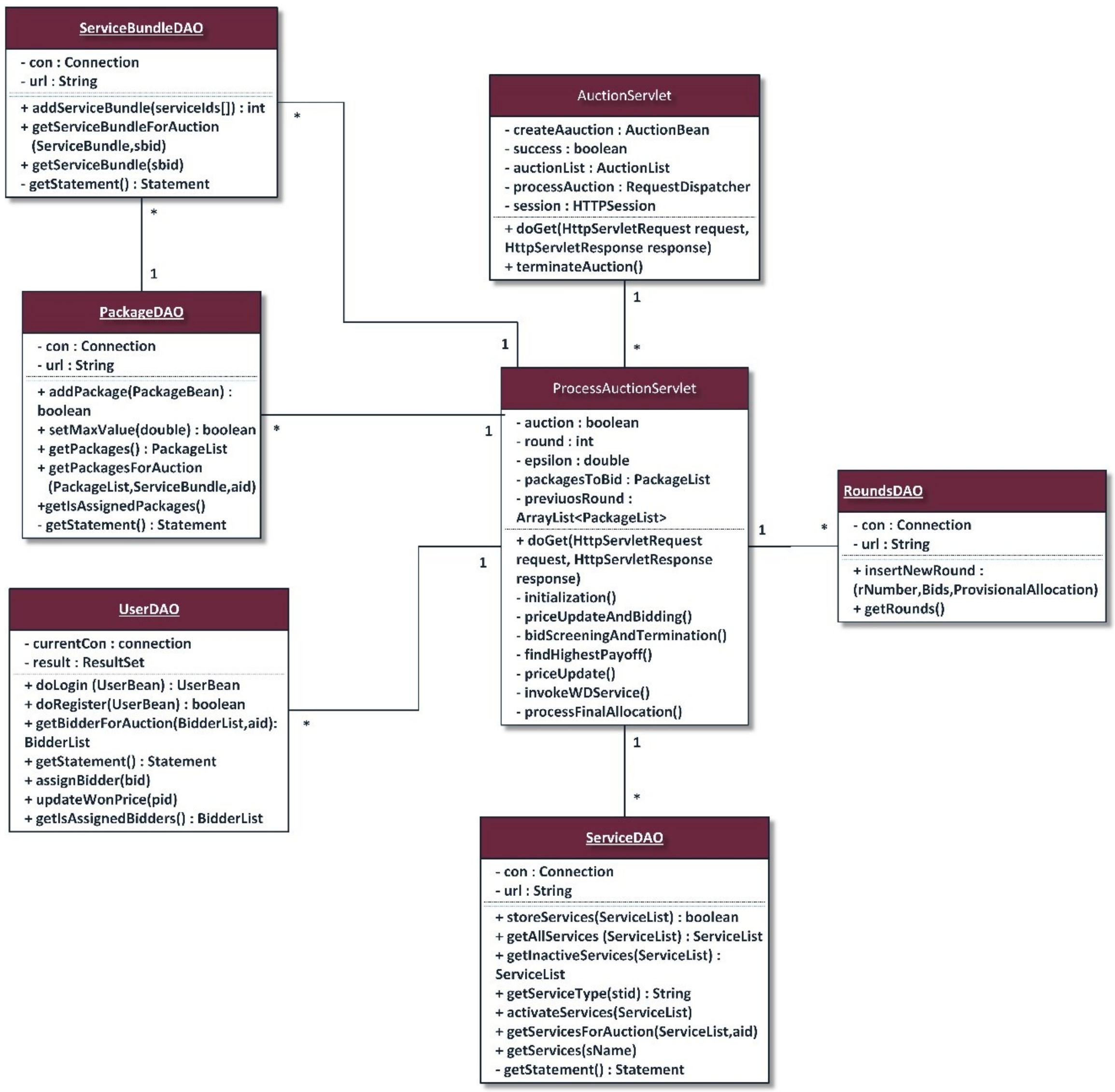

Figure 5-12: UML Class Diagram Showing Data Preparation for Winner Determination as Input

5.1.4 Data Preparation and Auction Process

Figure 5-12 shows the classes that are used by the Auction Control component for an auction process. These classes work together in order to provide data as an input for the Winner Determination class. Data is prepared by the ProcessAuctionServlet which is a Java Servlet class responsible for using other classes in order to obtain data from the database. Since we designed the auction system using Data Access Object (DAO) pattern, we have designed DAO classes in order to access each table of our database. For example, ServiceDAO is responsible for retrieving services from the database. There are different methods provided for this purpose, whose use follows from the ProcessAuctionServlet's needs. Other DAO classes such as UserDAO, RoundsDAO, PackageDAO, and ServiceBundleDAO function in the same way. They are designed to serve the ProcessAuctionServlet with any data-related jobs. The AuctionServlet class, on the other hand, is designed to receive the Manager's request for terminating an auction so that this Servlet can run or process the auction using the ProcessAuctionServlet. To better understand the auction process, the sequence diagram illustrated in Figure 5-13 shows the order of steps, starting from gathering data, moving to computing the Winner Determination, and ending in the termination of bidding.

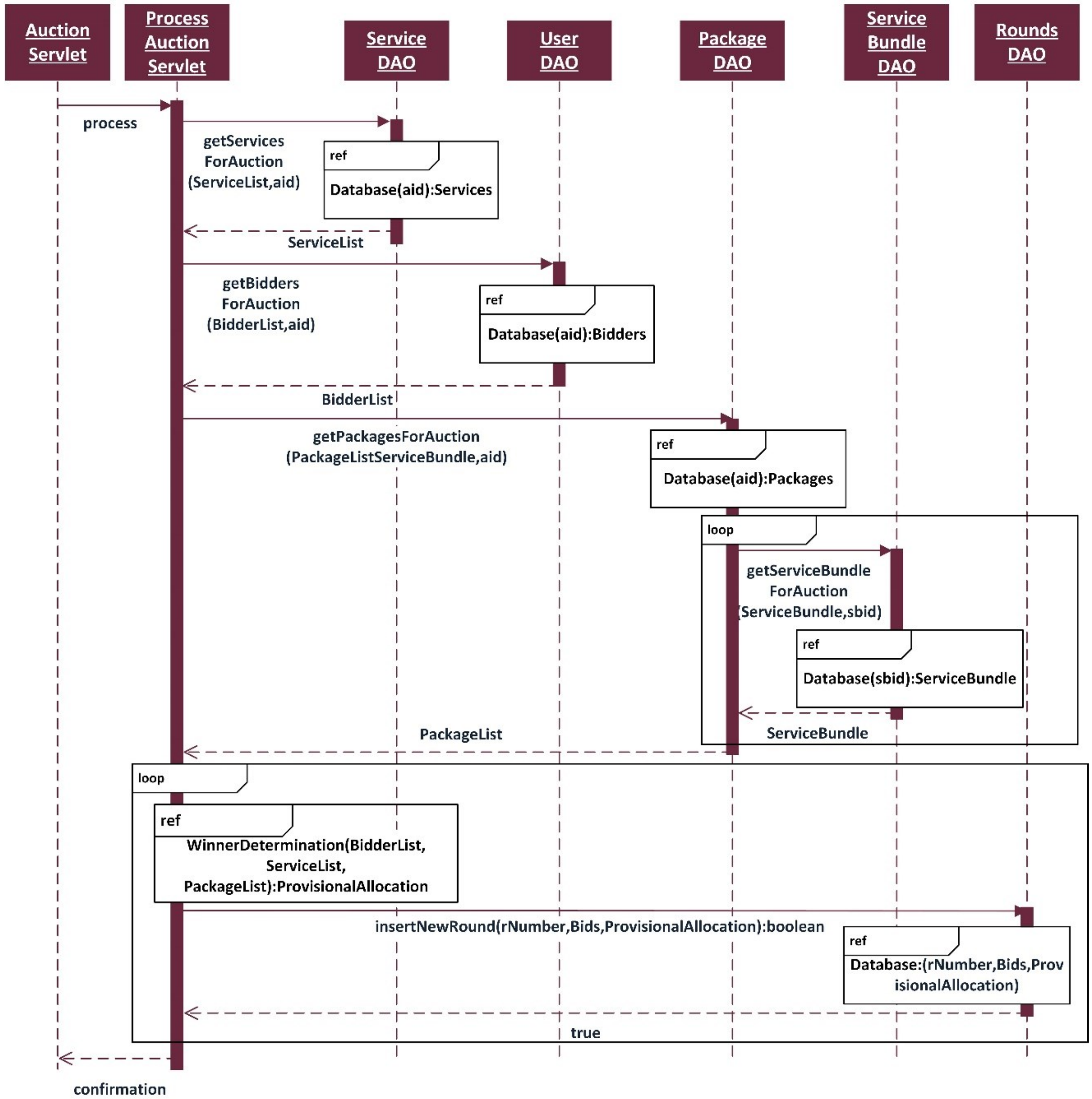

Figure 5-13: UML Sequence Diagram Illustrating the Sequence of Steps for an Auction Procedure

When the Manager clicks on the “Terminate” button in the Internet browser, the AuctionServlet invokes the ProcessAuctionServlet to process the auction. The ProcessAuctionServlet will then retrieve the required data from the database using the corresponding DAOs. The data that is retrieved from the database will be created as corresponding objects and will be held in the ProcessAuctionServlet until the bidding is terminated. As illustrated above, WinnerDetermination is invoked many times. This is because an auction contains several rounds and the provisional allocation of each round must be obtained. The results such as the submitted bids, provisional allocation, provider revenue, and customer value of each round of auction will be stored in the database as well. The steps of our auction's iterative procedure (described in section 3.2) are all written as methods in the ProcessAuctionServlet. After each round of the auction, the required object will be updated based on the provisional allocation. When the bidding is terminated a confirmation will be sent to the Manager so that he or she will be able to view the results.

5.2 System Implementation

In addition to the web-based auction system described at the beginning of this chapter, we also designed and implemented two other applications: the Data Generator and the Optimal Finder. In this section, we will describe the implementation of each of these systems.

5.2.1 Web-Based Auction System

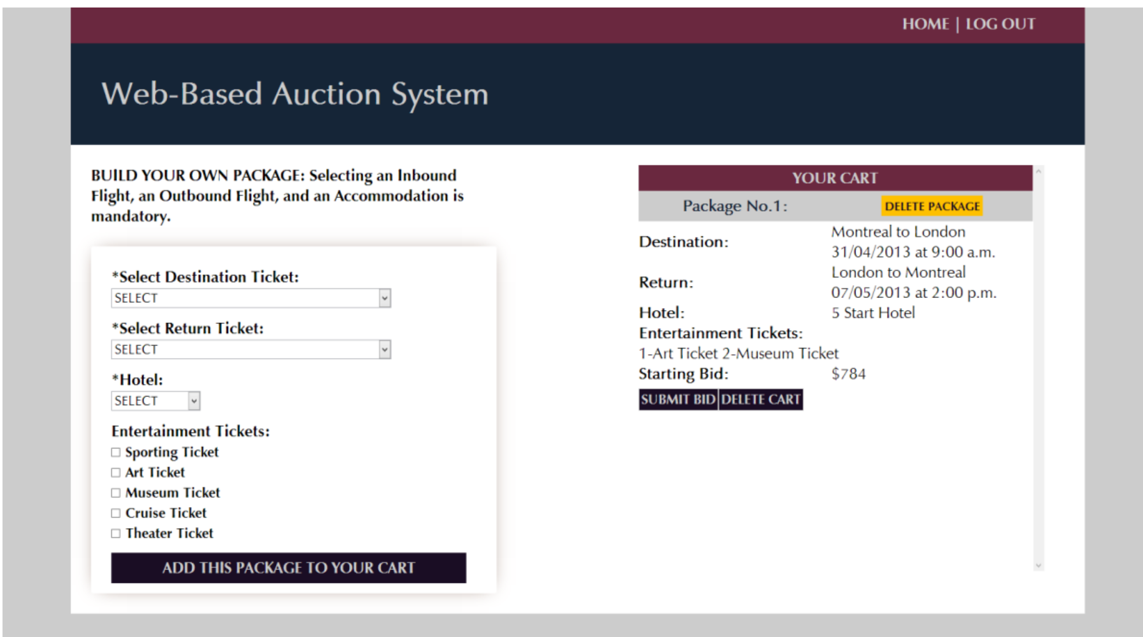

Figure 5-14: Adding Packages to Cart User Interface

We implemented our web-based auction system using J2EE technologies. We used Java Server Pages (JSP) in order to dynamically generate the auction system's web pages (see Figure 5-14 and 5-15).

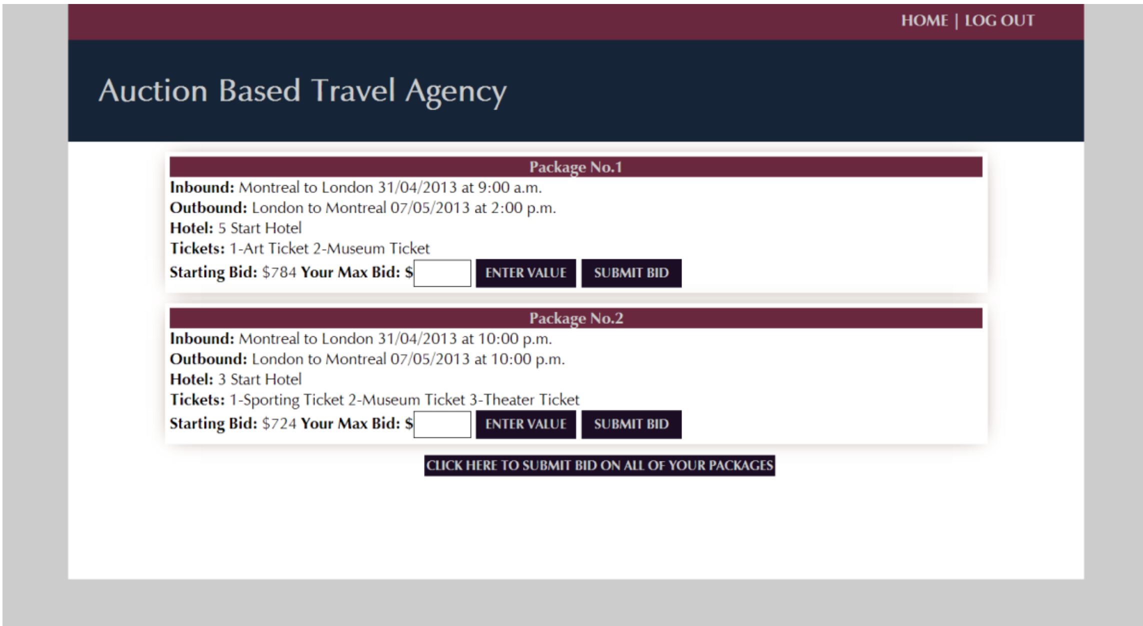

Figure 5-15: Submitting Value and Bid User Interface

We used Java Servlets in order to receive and respond to requests from JSPs across HTTP. For example, the Manager's request for receiving travel services was requested by JSP from these Servlets (see Figure 5-16).

We implemented the Travel, Hotel, and Entertainment Tickets web services using Java SOAP web services. The coding was conducted in NetBeans IDE 7.3 while the server in which our web-based auction system and web services were deployed was GlassFish 3+. For the purposes of our project, NetBeans proved to be the most efficient IDE, having sufficient functionalities for implementing both our Web Application and Web Services.

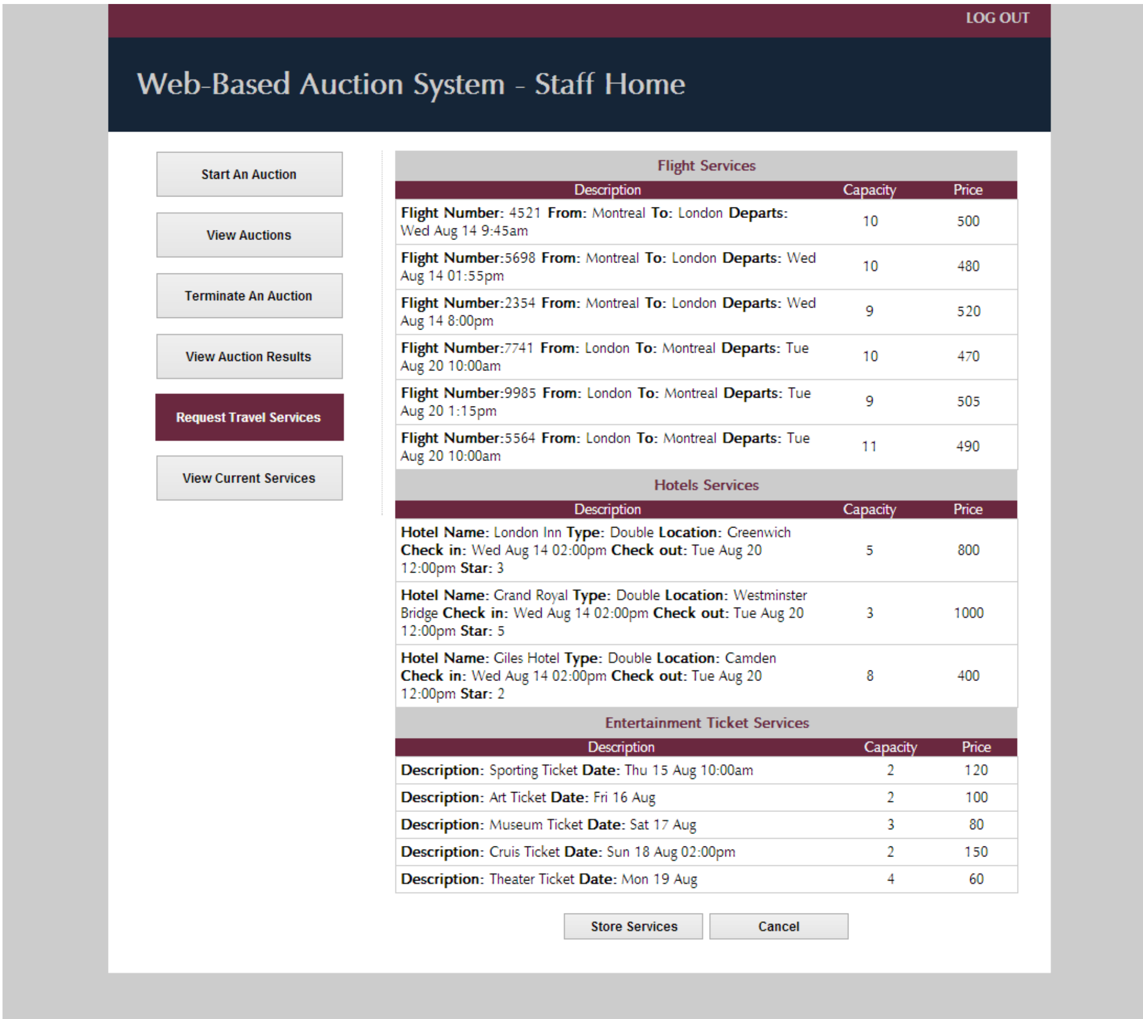

Figure 5-16: Request Travel Services User Interface

In order to design our web-based auction system, we used a DAO design pattern. This design pattern acts as an adapter between a component and the database. It encapsulates and abstracts all access to the database (ORACLE, 2002). Nevertheless, DAO has some advantages and disadvantages we had to contend with over the course of the project:

Advantages:

- A J2EE best practices.
- Separates two important parts of the system.
- The data source details can be hidden from other parts of the system.
- Acts as an intermediary between components of the system and the database.
- Reduces code duplication within an application for the accession and storage of data.
- Is designed for distributed architectures since data objects can be passed between different tiers.

Disadvantages:

- Requires large amount of codes to be written in each DAO class.

5.2.2 Data Generator & Optimal Finder

We also implemented a Data Generator application to randomly generate large data in order to solve our problem model. An Optimal Finder application was also implemented for the purposes of finding the optimal auction value and revenue so that we could compare them with the results that we obtained from our web-based auction system application. These two applications were also coded in Java. MySQL was used as our relational database management system as the data source. Storing and obtaining data from the MySQL database was performed on all three applications (the Web-Based and Data Generator Auction Systems as well as the Optimal Finder). In order to solve the iterative bidding model and the travel service customization model used in the Optimal Finder application we employed an IBM OPL interface for Java. As previously mentioned, OPL is a modeling language designed for combinatorial optimization that solves and simplifies optimization problems (IBM, 2009). This interface enables developers to integrate OPL models in Java applications. In order to use the functionalities of this interface in our applications we have added the oplall.jar into our project library. This library provides all the capabilities of OPL for Java applications. All three applications were deployed on a PC with 64-bit operating system on Windows 8, with a 4GB memory and 2.00G Inter CPU.

Chapter 6

Test Problems and Results

As our web-based auction system was not deployed on an Internet server which is open to the public, we were not able to recruit customers to interact with our system and produce sufficient data for analysis. Accordingly, we designed and implemented a random data generator (as described in section 5.2.2) in order to produce different numbers of customers, packages, and values as inputs for our web-based auction system. These results were then compared against the optimal results computed by our Optimal Finder application. In this chapter, we first describe the design of the testing data that we used to run different auctions, and we then elaborate on the experimental results.

6.1 Design of the Testing Data

We designed the data generator system to produce data in a way that respects all the required conditions. For example, each customer could not have more than 5 packages and each package had to contain the base configuration of services.

Consequently, the data generator randomly produced between 1 to 5 packages for each customer. The base configuration of each package had to contain one of DT (Destination Ticket), one of RT (Return Ticket), and one of HL (Hotel) in all instances.

Each service has a reservation price and a retail price. For the test, the reservation price of each service was set at 50% of the retail price. In this case, the other 50% was reduced from the retail price as discount. The reservation price for a package is calculated as the sum of the reservation prices for the services included in that package. A customer value was then added to each package, assigned randomly for the purposes of the test between the reservation price and the retail price.

This customer value was generated on the basis of common pricing schemes found in other online travel auctions such as Luxury Link (http://www.luxurylink.com), eBay Travel (http://www.ebay.com), and Sky Auction (http://skyauction.com). In these websites there is a "Buy It Now" option that enables customers to instantly buy a package at its retail price. However, if customers want to participate in an auction, they may be able to buy that package at a reservation price as low as 50% of the retail price. Using our random data generator, we created data for 10 groups of customers ranging from 100 to 1000 under 3 different configurational levels. For each group, five instances were randomly generated. The service capacity of each group was different and was allocated in proportion to the number of customers in a way that almost 85-90% of customers under Base-Config#1 would obtain a feasible package.

6.2 Experimental Results

Based on the test data, our web-based auction system is evaluated in terms of its revenue, value and runtime performance under different levels of service customization provided by the Auctioneer. In this study we have provided three levels of service customizability with different base configurations. The base configuration are as follows:

Table 6-1: Three Levels of Configurations

Services	Base-Config#1	Base-Config#2	Base-Config#3
DT (Destination Ticket)	One	One	One
RT (Return Ticket)	One	One	One
HL (Hotel)	One	One	One
ET (Entertainment Ticket)		Two	Four
Total	3	5	7

As expressed in Table 6-1, the Base-Config#1 has 3 services, the Base-Config#2 has 5 services and the Base-Config#3 has 7 services out of a total 14 possible services (see Appendix A). For the purposes of this study, we will use the results obtained under Base-Config#1 as the baseline for comparisons. For all instances of each group of Base-Config#1 the optimal value has been calculated using the Optimal Finder which uses the customization model presented in section 3.1.

Table 6-2: Optimal Value, Customer Value, and Provider Revenue under Base-Config#1

Group	Base-Config#1		
	(1) Optimal Value	(2) Auction Value	(3) Auction Revenue
1	$214,062	$212, 561	$177, 552
2	$440,020	$438, 023	$363, 508
3	$676,370	$658, 215	$547, 157
4	$868,295	$866, 210	$748, 825
5	$1,116,406	$1, 103, 153	$917, 645
6	$1,338,063	$1, 317, 755	$1, 098, 227
7	$1,547,420	$1, 527, 236	$1, 272, 707
8	$1,739,800	$1, 730, 734	$1, 440, 779
9	$1,984,359	$1, 967, 074	$1, 637, 937
10	$2,192,883	$2, 179, 908	$1, 814, 281

The results obtained from the auction system, which include the auction value and the auction revenue, are compared against the optimal ones computed by the Optimal Finder system in the above table. The first column of Table 6-2 shows the average optimal value of each group under Base-Config#1. Columns 2 and 3 show the solution value and revenue computed by our web-based auction system. During the auction procedure, epsilon ε for all bidding was set at 20 and all customers were assumed to use final-bid-repeating. As the results show, the average customer value computed in our web-based auction system achieves 96% of the optimal value across 5 instances of the 10 groups. Additionally, the average auction revenue obtained is almost 78% of its optimal value.

Additionally, Figure 6-1 illustrates the average auction value and revenue against the optimal value graphically.

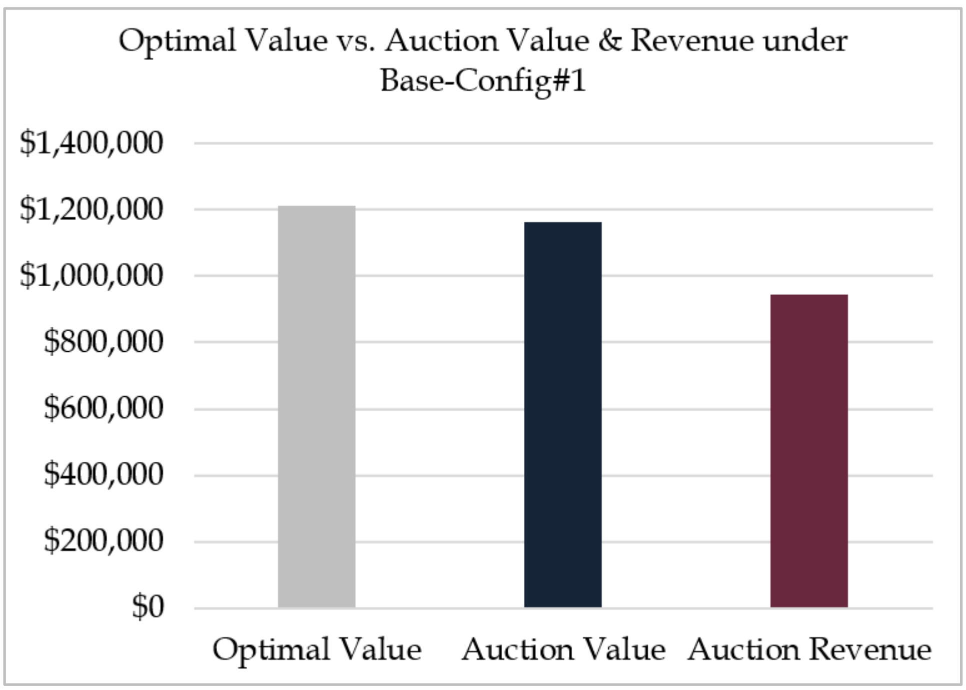

Figure 6-1: Optimal Value versus Auction Value and Auction Revenue under Base-Config#1

Table 6-3: Customer Value, and Provider Revenue at Different Level of Customizability

Group	Base-Config#1		Base-Config#2		Base-Config#3	
	(1) Auction Value	(2) Auction Revenue	(3) Auction Value	(4) Auction Revenue	(5) Auction Value	(6) Auction Revenue
1	$212, 561	$177, 552	$165, 371	$130,451	$110,588	$97,303
2	$438, 023	$363, 508	$344, 318	$259,538	$238,423	$199,205
3	$658, 215	$547, 157	$506, 749	$416,864	$352,880	$297,699
4	$866, 210	$748, 825	$673, 911	$512,075	$474,676	$386,933
5	$1, 103, 153	$917, 645	$862, 845	$655,209	$600,203	$495,656
6	$1, 317, 755	$1, 098, 227	$1,012,053	$764,373	$727,393	$595,221
7	$1, 527, 236	$1, 272, 707	$1,185,176	$911,181	$833,889	$679,650
8	$1, 730, 734	$1, 440, 779	$1,325,750	$1,057,472	$931,021	$789,554
9	$1, 967, 074	$1, 637, 937	$1,502,814	$1,185,776	$1,066,039	$874,574
10	$2, 179, 908	$1, 814, 281	$1,683,002	$1,324,203	$1,155,338	$972,554

In order to evaluate the effects of package customizability on customer value we solved the testing problems (see Appendix A) with Base-Config#2 and Base-Config#3. When conducting our iterative bidding procedure, all packages which did not contain the configurational requirements of Base-Config#2 and Base-Config#3 were excluded at the beginning of the bidding procedure. Columns three and four of Table 6-3 show the solution values and revenues under Base-Config#2. As can be seen, the average solution value is decreased to 23% of that with Base-Config#1 and the solution revenue is decreased to 28% of that in Base-Config#1. If Base-Config#3 is applied, the average solution value will decrease to 45% of that with Base-Config#1 and the solution revenue with decrease to 46% of that with

Base-Config#1. From the experimental results it is evident that reducing service customizability can decrease the overall customer value as well as providers' revenue. Figure 6-2 illustrates the decrease graphically.

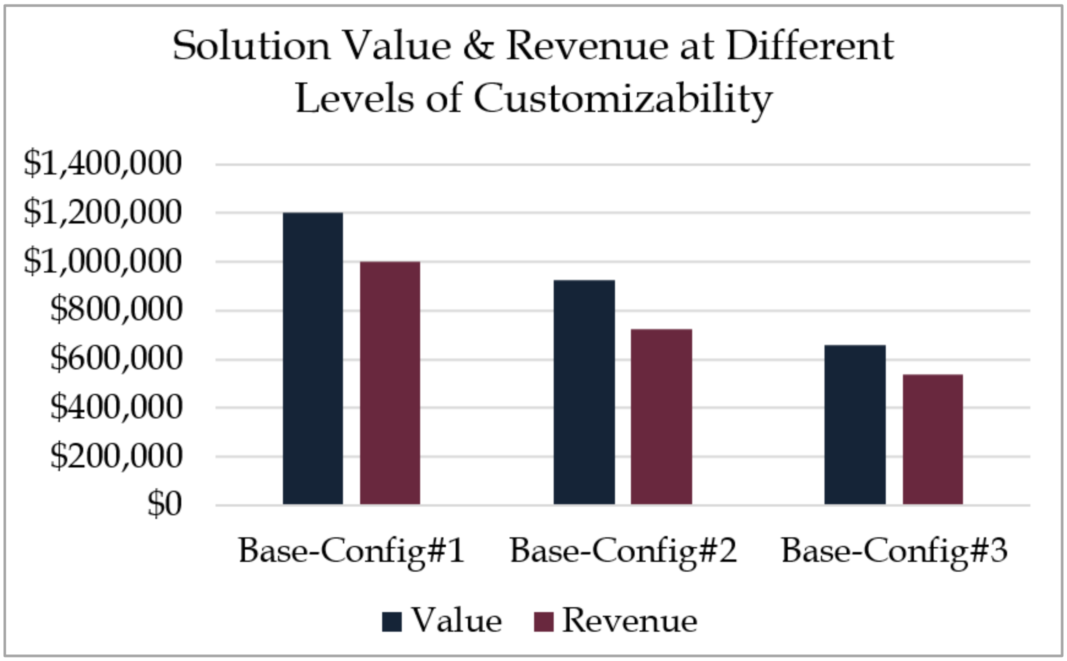

Figure 6-2: Solution Value & Revenue at Different Levels of Customizability

The average run time of our web-based auction system is also evaluated under the three base configuration levels with the Base-Config#1 used as the baseline for comparison. As Figure 6-3 illustrates, Base-Config#1 produced the largest average run time of 38:55. Base-Config#2 and Base-Config#3 rated second and third, respectively, with average run times of 19:35 and 7:25. From these results, it is evident that a reduction of service customizability will result in decreases to the average run time of the auction system.

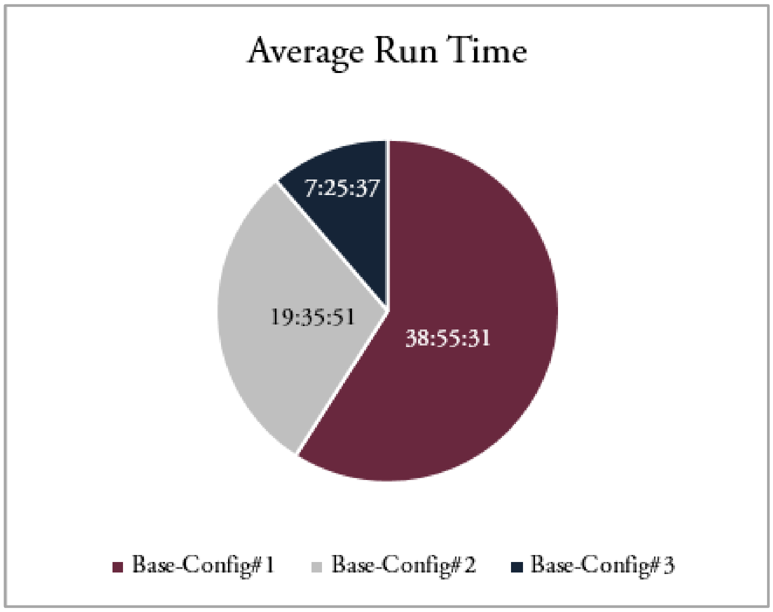

Figure 6-3: Average Run Time of Three Levels of Configurations

By reviewing the above tables and charts we can conclude from our experimental results that reductions to the customizability of services in the auction system will result in decreases to overall customer values and the auction revenues. If the Auctioneer provides more mandatory services, less revenue will follow. In terms of auction run time, the average runtime will decrease in the case of reductions to service customizability.

Chapter 7

Summary and Conclusions

In this chapter, we begin by summarizing the main contributions of our research. We then highlight our conclusions and offer some suggestions on further research directions.

The concept of mass customization has been the subject of a lot of attention to the manufacturing and service industries in recent years. Mass customization provides customers with the ability to select a number of available services and customize them as packages in the way that best suits their needs. The aim of this thesis was to design a web-based auction system for the mass customization of services integrated with an iterative bidding procedure. Following from this goal, we analysed the necessary requirements for our auction system. Most importantly, the system had to enable Bidders to submit bids, facilitate a Manager's control over the system, and utilize three web services to provide travel services such as flight tickets, hotels, and entertainment tickets. As our main contribution to this thesis, we designed and implemented a web-based auction system using J2EE technologies and web services. Our web-based auction system was designed in such a way that the Auctioneer provides a set of travel services that customers can then use to select, customize, and bid on travel services as several packages through a user friendly GUI.

We realized our system through a combination of problem solving and design. Since the customization problem model described in Chapter 3 was necessary to compute the provisional allocation of each round, we integrated the IBM ILOG OPL interface for Java in order to achieve this functionality in our system. In addition to the web-based auction system, we also designed and implemented two other Java applications named the Optimal Finder and the Data Generator. As we did not deploy our web-based auction system on an Internet server and we needed to simulate the effects of a large number of customers on the system, we designed the Data Generator to imitate customers and their travel packages. In order to evaluate our study we designed a set of testing data using our Data Generator application for the purposes of experimentation. We randomly generated 10 groups of customers ranging from 100 to 1000. For each group of customers under we then found the optimal solution using our Optimal Finder application.

Experimental results confirmed that the customization solutions computed by our web-based auction system were very close to optimal. It is also evident that reductions to the available levels of customization will decrease both the overall customer value and the providers' revenue under the same group of customers and packages. The average run time also decreases as the levels of customization reduce. Based on these results, we can conclude that the auction procedures in the model are capable of dealing with service customization problems on a large scale.

A number of avenues remain open for further investigation. First of all, for the purposes of this research, the capacity of services were fixed and known during the auction. In the future, researchers could consider studying this problem in settings wherein the providers' service capacities could be expanded in a real-time manner or in a dynamic environment that uses the same web services. Additionally, since the current system was deployed on a virtual server and bid on behalf of its customers, one direction for further development would be to deploy the web application on a physical server with public availability, so that customers could use the system to manually submit their bids during an auction. In this case customers would be able to rebid if they are out-bided.

Bibliography

Bykowsky, M. M., Cull, R. J., & Ledyard, J. O. (2000). Mutually destructive bidding: The FCC auction design problem. *Journal of Regulatory Economics*.

Cassady , J. R. (1967). *Auctions and Auctioneering.* Berkeley: University of California Press.

Chockshi, N., & McFarlane, D. (2008). A distributed architecture for reconfigurable control of continuous process operations. *Journal of Intelligent Manufacturing*, 19, 215-232.

Cramton, P., Shoham, Y., & Steinberg, R. (2006). Introduction to Combinatorial Auctions.

Da Silveira, G., Borenstein, D., & Fogliatto, F. S. (2001). Mass customization: Literature review and research directions. *International Journal of Production Economics*, 72, 1-13.

Davis, S. M. (1987). *Future perfect.* Reading, Mass: Addison-Wesley.

deVries, S., & Vohra, R. V. (2003). Combinatorial auctions: A survey. *INFORMS Journal on Computing*, 15, 284-309.

Du et al. (2001). Architecture of product family: Fundamentals and methodology. *Concurrent Engineering: Research and Application*, 9, 309-325.

Fiore, A. M., Lee, S. E., & Kunz, G. (2003). Psychographic variables affecting willingness to use body-scanning. *Journal of Business and Management*, 9, 271-287.

Guo, L., Hasegawa, T., Luh, P. B., Tamura, S., & Oblak, J. M. (1994). Holonic planning and scheduling for a robotic assembly testbed. *Proceedings of the 4th international conference on CIM and automation technology*, 142- 149.

Hailu, A., & Thoyer, S. (2006). Multi-unit auction format design. *Journal of Economic Interaction and Coordination*, 1, 129-146.

Hoa, H. (2003). *What is Service-Oriented Architecture*. Retrieved 01 03, 2012, from O'Reilly xml.com: http://www.xml.com/pub/a/ws/2003/09/30/soa.html

IBM. (2009). *What is OPL?* Retrieved 06 21, 2011, from IBM: http://pic.dhe.ibm.com/infocenter/odmeinfo/v3r3/index.jsp?topic=/ilog.odms.ide.odme.help/Content/Optimization/Documentation/ODME/_pubskel/ODME_pubskels/startALL_ODME_Eclipse_and_Xplatform_

Jiao, J., & Tseng, M. M. (1999). A methodology of developing product family architecture for mass customization. *Journal of Intelligent Manufacturing*, 10, 3-20.

Mass-Colell, A., Whinstom, M. D., & Green, J. R. (1995). *Microeconomics.* Oxford: Oxford University Press.

Menesez, F. M., & Monteiro, P. K. (2005). *An introduction to auction theory.* New York: Oxford Unversity Press.

Meyer, M. H., & Utterback, J. M. (1993). The product family and the dynamics of core capability. *Sloan Management Review*, 29-47.

Moses, S., Gruenwald, L., & Dadachanji, K. (2008). A scalable data structure for real-time estimation of resource availability in build-to-order environments. *Journal of Intelligent Manufacturing*, 19, 611-622.

Narumanchi, M., & Vidal, J. (2006). Algorithms for distributed winner determination in combinatorial auctions. *Agent-Mediated Electronic Commerce. Designing Trading Agents and Mechanisms*, 3937, 43-56.

ORACLE. (2002). *Core J2EE Patterns - Data Access Object*. Retrieved from ORACLE: http://www.oracle.com/technetwork/java/dataaccessobject-138824.html

Papazoglu, M. P. (2008). *Web Services: Principles and technology.* England: Pearson Education Limited.

Parkes & Kalagnanam. (2005). Models for Iterative Multiattribute Procurement Auctions. *Management Science*, 435-451.

Parkes, & Kalagnanam. (2005). Models for iterative multiattribute procurement auctions. *management Science*, 51, 435-451.

Parkes, D. C., & Ungar, L. (2001). An Auction-based method for Decentralized train scheduling. *Proceedings of 5th International Conference on Autonomous Agents*, 43-50.

Parkes, D. C., & Ungar, L. H. (2000). Iterative Combinatorial Auctions: Theory and Practice. *Proceedings of the Seventeenth National Conference on Artificial*

Inteligence and Twelfth Conference on Innovative Applications of Artificial Inteligence, 74-81.

Porter, D., Rassenti, S., Roopnarine, A., & Smith, V. (2003). Combinatorial auction design. *Proceeding of the National Academy of Science of the United States*, 100, 11153-11157.

Salvador, F., de Holan, P. M., & Piller , F. (2009). Cracking the code of mass customization. *MIT Sloan Management Review*, 71-78.

Sampson, S. E. (2001). *Understanding service businesses: Applying principles of the unified services theory (2nd ed.)*.

Shen, W., Wang, L., & Qi, H. (2006). Agent-based distributed manufacturing process planning and scheduling: A state-of-the-art survey. *IEEE Transactions on Systems, Man, and Cybernetics,Part C: Applications and Reviews*, 36, 563–577.

Simpson et al. (2005). CABOB: A fast optimal algorithm for winner determination in combinatorial auctions. *Management Science*, 51, 374–390.

Tenorio, R. (1993). Revenue Equivalence and Bidding Behavior in a Multi-Unit Auction Market: An Empirical Analysis. *Review of Economics and Statistics*, 75, 302-314.

Tseng & Du. (1998). Design by customers for mass customization products. *CIRP Annals Manufacturing Technology*, 47, 103-106.

Tseng, M. M., & Jiao, J. (2001). Mass customization. In G. Salvendy (Ed.). *Handbook of Industrial Engineering: Technology and Operations Management, (3rd ed.)*.

Ulrich, K. (1995). The role of product architecture in the maufacturing film. *Research Policy*, 24, 419-440.

Vickery, W. (1961). Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16, 8-37.

W3C. (2004). *Web Services Architecture*. Retrieved 02 16, 2011, from W3C.

Appendix A: Detailed Experimental Results

The following table shows the capacities of each service used for the three base configurations.

	Capacity for all groups									
Service	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
DT1	40	80	120	160	200	240	280	320	360	400
DT2	30	60	90	120	150	180	210	240	270	300
DT3	30	60	90	120	150	180	210	240	270	300
RT1	30	60	90	120	150	180	210	240	270	300
RT2	30	60	90	120	150	180	210	240	270	300
RT3	40	80	120	160	200	240	280	320	360	400
HL1	20	40	60	80	100	120	140	160	180	200
HL2	40	80	120	160	200	240	280	320	360	400
HL3	30	60	90	120	150	180	210	240	270	300
ET1	30	60	90	120	150	180	210	240	270	300
ET2	30	60	90	120	150	180	210	240	270	300
ET3	40	80	120	160	200	240	280	320	360	400
ET4	30	60	90	120	150	180	210	240	270	300
ET5	50	100	150	200	250	300	350	400	450	500

The following tables illustrate detailed information under Base-Config#1:

Base-Config#1 - Group 1 - 100 Customers					
Instance	1	2	3	4	5
Capacity			C1		
Rounds	109	130	133	112	93
Run Time (Minutes)	2:59	3:10	3:16	3:04	2:53
Auctioneer Revenue	$168,574	$185,655	$181,943	$176,855	$174,732
Customer Value	$202,396	$223,243	$217,694	$210,441	$209,030
Optimal Value	$203,896	$224,749	$219,194	$211,941	$210,530
Allocated Customers	78	88	87	83	83

Base-Config#1 - Group 2 - 200 Customers					
Instance	1	2	3	4	5
Capacity			C2		
Rounds	91	62	113	96	64
Run Time (Minutes)	6:13	5:26	7:42	7:22	5:34
Auctioneer Revenue	$362,503	$366,142	$348,537	$373,690	$366,667
Customer Value	$436,387	$444,738	$419,022	$449,209	$440,757
Optimal Value	$438,352	$446,249	$421,194	$451,360	$442,947
Allocated Customers	174	175	164	180	176

Base-Config#1 - Group 3 - 300 Customers					
Instance	1	2	3	4	5
Capacity			C3		
Rounds	88	108	80	126	128
Run Time (Minutes)	12:17	13:53	10:11	14:38	12:35
Auctioneer Revenue	$548,219	$544,323	$539,323	$546,823	$557,096
Customer Value	$660,060	$655,342	$647,383	$657,221	$671,069
Optimal Value	$678,127	$673,569	$665,725	$675,314	$689,114
Allocated Customers	263	260	257	261	267

Base-Config#1 - Group 4 - 400 Customers					
Instance	1	2	3	4	5
Capacity			C4		
Rounds	110	110	132	113	97
Run Time (Minutes)	14:52	15:59	17:33	17:11	16:30
Auction Revenue	$721,925	$718,049	$716,578	$720,790	$718,679
Auction Value	$864,197	$864,328	$870,271	$868,896	$863,357
Optimal Value	$866,214	$866,549	$872,362	$870,793	$865,555
Allocated Customers	344	343	341	343	342

Base-Config#1 - Group 5 - 500 Customers					
Instance	1	2	3	4	5
Capacity			C5		
Rounds	110	155	150	117	153
Run Time (Minutes)	32:10	39:06	43:10	36:04	39:01
Auction Revenue	$936,142	$908,258	$927,868	$895,263	$920,696
Auction Value	$1,124,785	$1,094,753	$1,116,278	$1,076,112	$1,103,837
Optimal Value	$1,114,821	$1,113,943	$1,135,365	$1,094,355	$1,123,546
Allocated Customers	450	434	447	427	441

Base-Config#1 - Group 6 - 600 Customers					
Instance	1	2	3	4	5
Capacity			C6		
Rounds	136	147	123	126	133
Run Time (Minutes)	35:04	46:02	30:59	51:50	47:15
Auction Revenue	$1,096,123	$1,085,110	$1,113,263	$1,102,426	$1,094,214
Auction Value	$1,309,051	$1,308,551	$1,338,327	$1,318,758	$1,314,087
Optimal Value	$1,329,245	$1,327,649	$1,359,369	$1,339,854	$1,334,196
Allocated Customers	524	516	535	528	523

Base-Config#1 - Group 7 - 700 Customers					
Instance	1	2	3	4	5
Capacity			C7		
Rounds	128	166	107	105	132
Run Time (Minutes)	54:46	66:56	49:54	54:42	57:36
Auction Revenue	$1,272,034	$1,266,555	$1,280,065	$1,269,907	$1,274,976
Auction Value	$1,527,319	$1,516,661	$1,526,490	$1,527,867	$1,537,842
Optimal Value	$1,547,625	$1,537,489	$1,546,502	$1,546,799	$1,558,687
Allocated Customers	611	604	614	608	612

Base-Config#1 - Group 8 - 800 Customers					
Instance	1	2	3	4	5
Capacity			C8		
Rounds	125	124	126	129	132
Run Time (Minutes)	64:29	69:21	65:57	58:26	76:09
Auction Revenue	$1,461,727	$1,452,090	$1,422,528	$1,429,398	$1,438,152
Auction Value	$1,751,086	$1,739,294	$1,714,067	$1,718,488	$1,730,735
Optimal Value	$1,760,111	$1,748,319	$1,723,174	$1,727,555	$1,739,843
Allocated Customers	702	694	681	682	687

Base-Config#1 - Group 9 - 900 Customers					
Instance	1	2	3	4	5
Capacity			C9		
Rounds	109	123	106	179	155
Run Time (Minutes)	71:18	66:08	56:45	67:45	90:54
Auction Revenue	$1,624,128	$1,641,589	$1,630,618	$1,654,822	$1,638,528
Auction Value	$1,954,353	$1,976,458	$1,952,136	$1,988,059	$1,964,366
Optimal Value	$1,971,638	$1,993,743	$1,969,421	$2,005,344	$1,981,651
Allocated Customers	776	788	781	795	779

Base-Config#1 - Group 10 - 1000 Customers					
Instance	1	2	3	4	5
Capacity			C10		
Rounds	155	164	105	117	129
Run Time (Minutes)	75:11	109:12	55:52	55:47	85:09
Auction Revenue	$1,839,165	$1,794,337	$1,824,342	$1,808,854	$1,804,709
Auction Value	$2,213,673	$2,151,892	$2,191,293	$2,167,750	$2,174,930
Optimal Value	$2,226,648	$2,164,867	$2,204,268	$2,180,725	$2,187,905
Allocated Customers	882	856	876	863	869

Following tables show detailed information under Base-Config#2:

Base-Config#2 – Group 1 - 100 Customers					
Instance	1	2	3	4	5
Capacity			C1		
Rounds	85	105	105	87	72
Run Time (Minutes)	1:47	1:38	1:43	1:44	1:10
Auction Revenue	$114,630	$137,385	$138,277	$134,410	$127,554
Auction Value	$157,869	$171,897	$171,978	$166,248	$158,863
Allocated Customers	58	66	64	59	61

Base-Config#2 – Group 2 - 200 Customers					
Instance	1	2	3	4	5
Capacity	C2	C2	C2	C2	C2
Rounds	76	48	92	74	53
Run Time (Minutes)	2:36	2:39	3:18	3:41	3:00
Auction Revenue	$279,127	$248,977	$247,461	$272,794	$249,334
Auction Value	$344,746	$351,343	$326,837	$354,875	$343,790
Allocated Customers	154	152	141	158	154

Base-Config#2 – Group 3 - 300 Customers					
Instance	1	2	3	4	5
Capacity			C3		
Rounds	70	89	62	98	99
Run Time (Minutes)	7:14:50	6:48:10	4:28:50	8:20:28	7:10:21
Auction Revenue	$400,200	$424,572	$420,672	$421,054	$417,822
Auction Value	$501,646	$511,167	$511,433	$499,488	$510,012
Allocated Customers	192	190	188	191	195

Base-Config#2 – Group 4 - 400 Customers					
Instance	1	2	3	4	5
Capacity			C4		
Rounds	90	92	106	90	76
Run Time (Minutes)	7:34:55	8:57:02	7:43:19	8:14:53	6:45:54
Auction Revenue	$505,348	$531,356	$487,273	$511,761	$524,636
Auction Value	$656,790	$674,176	$670,109	$686,428	$682,052
Allocated Customers	255	257	256	254	250

Base-Config#2 – Group 5 - 500 Customers					
Instance	1	2	3	4	5
Capacity			C5		
Rounds	85	121	123	90	130
Run Time (Minutes)	19:37:18	16:25:19	22:26:48	19:28:34	17:33:27
Auction Revenue	$692,745	$626,698	$630,950	$653,542	$672,108
Auction Value	$888,580	$853,907	$881,860	$817,845	$872,031
Allocated Customers	338	326	331	312	326

Base-Config#2 – Group 6 - 600 Customers					
Instance	1	2	3	4	5
Capacity	C6	C6	C6	C6	C6
Rounds	113	121	101	101	108
Run Time (Minutes)	16:07:50	20:42:54	13:00:47	29:01:36	21:44:06
Auction Revenue	$767,286	$748,726	$757,019	$804,771	$744,066
Auction Value	$994,879	$1,007,584	$1,030,512	$1,015,444	$1,011,847
Allocated Customers	383	387	391	396	382

Base-Config#2 – Group7 - 700 Customers					
Instance	1	2	3	4	5
Capacity			C7		
Rounds	102	131	86	81	107
Run Time (Minutes)	30:40:10	32:07:41	24:27:04	28:26:38	33:24:29
Auction Revenue	$954,026	$911,920	$870,444	$927,032	$892,483
Auction Value	$1,176,036	$1,167,829	$1,190,662	$1,176,458	$1,214,895
Allocated Customers	458	447	461	456	459

Base-Config#2 – Group8- 800 Customers					
Instance	1	2	3	4	5
Capacity			C8		
Rounds	103	99	97	104	106
Run Time (Minutes)	27:43:40	40:13:23	27:02:22	32:43:22	39:35:53
Auction Revenue	$1,096,295	$1,030,984	$1,038,445	$1,114,930	$1,006,706
Auction Value	$1,348,336	$1,321,863	$1,319,832	$1,306,051	$1,332,666
Allocated Customers	527	514	511	505	515

Base-Config#2 – Group9- 900 Customers					
Instance	1	2	3	4	5
Capacity			C9		
Rounds	89	96	88	138	129
Run Time (Minutes)	40:38:28	39:01:07	23:50:06	40:39:00	48:10:37
Auction Revenue	$1,185,613	$1,231,192	$1,157,739	$1,141,827	$1,212,511
Auction Value	$1,504,852	$1,502,108	$1,483,623	$1,510,925	$1,512,562
Allocated Customers	582	583	578	580	576

Base-Config#2 – Group10- 1000 Customers					
Instance	1	2	3	4	5
Capacity	C10	C10	C10	C10	C10
Rounds	119	133	82	92	103
Run Time (Minutes)	40:35:56	46:57:22	24:01:22	30:07:23	36:36:52
Auction Revenue	$1,250,632	$1,345,753	$1,422,987	$1,248,109	$1,353,532
Auction Value	$1,748,802	$1,678,476	$1,687,296	$1,647,490	$1,652,947
Allocated Customers	644	625	657	647	634

The following tables show detailed information under Base-Config#3:

Base-Config#3 – Group1- 100 Customers					
Instance	1	2	3	4	5
Capacity			C1		
Rounds	69	77	84	76	55
Run Time (Minutes)	0:30:26	0:30:24	0:43:07	0:31:17	0:31:08
Auction Revenue	$92,716	$103,967	$98,249	$93,733	$97,850
Auction Value	$103,222	$118,319	$115,378	$107,325	$108,696
Allocated Customers	34	38	37	37	37

Base-Config#3 – Group2- 200 Customers					
Instance	1	2	3	4	5
Capacity			C2		
Rounds	54	38	77	59	42
Run Time (Minutes)	1:14:36	0:55:25	1:04:41	1:01:53	1:13:29
Auction Revenue	$195,752	$197,717	$191,695	$205,530	$205,334
Auction Value	$240,013	$249,053	$217,891	$251,557	$233,601
Allocated Customers	75	74	69	76	77

Base-Config#3 – Group3- 300 Customers					
Instance	1	2	3	4	5
Capacity			C3		
Rounds	59	66	52	74	82
Run Time (Minutes)	2:42:08	3:11:35	1:37:46	2:38:02	2:38:33
Auction Revenue	$307,003	$293,934	$291,234	$284,348	$311,974
Auction Value	$349,832	$347,331	$343,113	$348,327	$375,799
Allocated Customers	110	112	111	115	117

Base-Config#3 – Group4- 400 Customers					
Instance	1	2	3	4	5
Capacity			C4		
Rounds	66	73	82	78	59
Run Time (Minutes)	2:22:43	3:02:13	3:51:40	3:15:53	3:37:48
Auction Revenue	$382,620	$373,385	$386,952	$396,435	$395,273
Auction Value	$483,950	$484,024	$461,244	$486,582	$457,579
Allocated Customers	148	147	143	147	147

Base-Config#3 – Group5 - 500 Customers					
Instance	1	2	3	4	5
Capacity			C5		
Rounds	74	99	92	74	106
Run Time (Minutes)	7:23:54	6:15:22	6:02:36	7:12:48	8:58:26
Auction Revenue	$524,240	$490,459	$510,327	$474,489	$478,762
Auction Value	$618,632	$580,219	$613,953	$581,100	$607,110
Allocated Customers	189	182	197	179	185

Base-Config#3 – Group6- 600 Customers					
Instance	1	2	3	4	5
Capacity			C6		
Rounds	86	97	82	82	85
Run Time (Minutes)	7:00:48	7:49:32	5:16:02	11:55:18	7:33:36
Auction Revenue	$613,829	$596,811	$601,162	$595,310	$568,991
Auction Value	$733,069	$732,789	$749,463	$685,754	$735,889
Allocated Customers	220	227	230	222	220

Base-Config#3 – Group7- 700 Customers					
Instance	1	2	3	4	5
Capacity	C7	C7	C7	C7	C7
Rounds	88	115	67	71	84
Run Time (Minutes)	12:35:47	14:43:31	11:28:37	11:29:13	9:12:58
Auction Revenue	$699,619	$658,609	$691,235	$673,051	$675,737
Auction Value	$824,752	$818,997	$854,834	$825,048	$845,813
Allocated Customers	257	260	264	261	257

Base-Config#3 – Group8- 800 Customers					
Instance	1	2	3	4	5
Capacity			C8		
Rounds	85	76	78	79	81
Run Time (Minutes)	9:40:21	11:05:46	13:11:24	12:51:19	15:59:29
Auction Revenue	$789,333	$813,170	$796,616	$743,287	$805,365
Auction Value	$910,565	$921,826	$908,456	$962,353	$951,904
Allocated Customers	309	291	300	300	302

Base-Config#3 – Group9- 900 Customers					
Instance	1	2	3	4	5
Capacity			C9		
Rounds	74	73	66	115	101
Run Time (Minutes)	13:32:49	9:15:31	9:04:48	13:33:00	19:59:53
Auction Revenue	$909,512	$870,042	$864,228	$877,056	$852,035
Auction Value	$1,094,438	$1,067,287	$1,034,632	$1,033,791	$1,100,045
Allocated Customers	326	331	344	342	343

Base-Config#3 – Group10- 1000 Customers					
Instance	1	2	3	4	5
Capacity	C10	C10	C10	C10	C10
Rounds	105	100	64	78	77
Run Time (Minutes)	15:02:12	20:44:53	12:17:26	10:02:28	12:46:21
Auction Revenue	$1,029,932	$986,885	$966,901	$940,604	$938,449
Auction Value	$1,151,110	$1,118,984	$1,183,298	$1,170,585	$1,152,713
Allocated Customers	379	368	385	362	382