# Software Architecture Document

## Online Catering Service 1.0

## Yummy Inc.

## SAD Revision History

| Date | Version | Description | Author |
|---|---|---|---|
| 2005-03-16 | 0.1 | Significant Use-Cases : the key requirements | Yummy Inc. Architecture Team |
| 2003-03-18 | 0.2 | Candidate architecture : the high level architecture of the system | Yummy Inc. Architecture Team |
| 2003-03-20 | 0.3 | Initial Deployment Model | Yummy Inc. Architecture Team |
| 2003-04-24 | 0.4 | Key abstractions : the key data elements used in the system | Yummy Inc. Architecture Team |
| 2003-04-29 | 0.5 | Analysis Model | Yummy Inc. Architecture Team |
| 2003-05-05 | 0.6 | Design Model | Yummy Inc. Architecture Team |
| 2004-05-11 | 0.7 | Concurrency mechanisms | Yummy Inc. Architecture Team |
| ????-??-?? | 0.8 |  | Yummy Inc. Architecture Team |

# Table of Contents

## 1. Introduction

This document comes as a complement to the article "Developing a J2EE Architecture with Rational Software Architect using the Rational Unified Process®" [RUPRSA]. It illustrates what can be the content of a Software Architecture Document (SAD) produced during the RUP Elaboration phase.

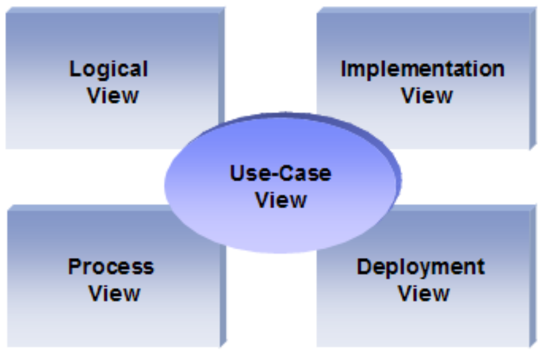
As stated in the companion article, a RUP Software Architect will typically perform height major steps in order to define a global architecture, and each time an activity is completed, a specific section of the SAD is enriched accordingly.

| Architectural activities | Software Architecture Document |
|---|---|
| Step 1 - Identify and prioritize significant Use-Cases | Section 4 |
| Step 2 - Define the candidate architecture | Section 3, 5.1, 10, 11 |
| Step 3 - Define the initial Deployment Model | Section 7 |
| Step 4 - Identify key abstractions | Section 9 |
| Step 5 - Create an Analysis Model | Section 5 |
| Step 6 - Create the Design Model | Section 5 |
| Step 7 - Document concurrency mechanisms | Section 6, 7 |
| Step 8 - Create the Implementation Model | Section 8 |

### 1.1 Purpose

The Software Architecture Document (SAD) provides a comprehensive architectural overview of the Online Catering Service 1.0 offered by Yummy Inc. It presents a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

In order to depict the software as accurately as possible, the structure of this document is based on the "4+1" model view of architecture [KRU41].



The "4+1" View Model allows various stakeholders to find what they need in the software architecture.

## 1.2 Scope

The scope of this SAD is to depict the architecture of the online catering application created by the company Yummy Inc.

## 1.3 Definitions, Acronyms and Abbreviations

**RUP**: Rational Unified Process

**UML:** Unified Modeling Language

**SAD:** Software Architecture Document

## 1.4 References

| | |
|---|---|
| [KRU41]: | The "4+1" view model of software architecture, Philippe Kruchten, November 1995, http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/Pbk4p1.pdf |
| [RSA]: | IBM Rational Software Architect http://www-306.ibm.com/software/awdtools/architect/swarchitect/index.html |
| [RUP]: | The IBM Rational Unified Process : http://www-306.ibm.com/software/awdtools/rup/index.html |
| [RUPRSA]: | Developing a J2EE Architecture with Rational Software Architect using the Rational Unified Process®, IBM DeveloperWorks, Jean-Louis Maréchaux, Mars 2005, http://www-128.ibm.com/developerworks/rational/library/05/0816_Louis/ |

## 1.5 Overview

In order to fully document all the aspects of the architecture, the Software Architecture Document contains the following subsections.

Section 2: describes the use of each view

Section 3: describes the architectural constraints of the system

Section 4: describes the functional requirements with a significant impact on the architecture

Section 5: describes the most important use-case realization. Will contain the Analysis Model and the Design Model

Section 6: describes design's concurrency aspects

Section 7: describes how the system will be deployed. Will contain the Deployment Model

Section 8: describes the layers and subsystems of the application

Section 9: describes any significant persistent element. Will contain the Data Model

Section 10: describes any performance issues and constraints

Section 11: describes any aspects related to the quality of service (QoS) attributes

## 2. Architectural Representation

This document details the architecture using the views defined in the "4+1" model [KRU41], but using the RUP naming convention. The views used to document the Yummy Inc. application are:

### Logical view

**Audience**: Designers.
**Area**: Functional Requirements: describes the design's object model. Also describes the most important use-case realizations.
**Related Artifacts**: Design model

### Process view

**Audience**: Integrators.
**Area**: Non-functional requirements: describes the design's concurrency and synchronization aspects.
**Related Artifacts**: (no specific artifact).

### Implementation view

**Audience**: Programmers.
**Area**: Software components: describes the layers and subsystems of the application.
**Related Artifacts**: Implementation model, components

### Deployment view

**Audience**: Deployment managers.
**Area**: Topology: describes the mapping of the software onto the hardware and shows the system's distributed aspects.
**Related Artifacts**: Deployment model.

### Use Case view

**Audience**: all the stakeholders of the system, including the end-users.
**Area**: describes the set of scenarios and/or use cases that represent some significant, central functionality of the system.
**Related Artifacts** : Use-Case Model, Use-Case documents

### Data view (optional)

**Audience**: Data specialists, Database administrators
**Area**: Persistence: describes the architecturally significant persistent elements in the data model
**Related Artifacts**: Data model.

# 3. Architectural Goals and Constraints

This section describes the software requirements and objectives that have some significant impact on the architecture

## 3.1 Technical Platform

The Yummy Inc online application will be deployed onto a J2EE application server (Websphere Application Server version 6, as it is already the application server use for internal applications).

## 3.2 Transaction

The Yummy Inc online application is transactional, leveraging the technical platform capabilities. Transaction management model of the J2EE platform will be reused intensively.

## 3.3 Security

The system must be secured, so that a customer can make online payments.

The application must implement basic security behaviors:

- Authentication: Login using at least a user name and a password
- Authorization: according to their profile, online customer must be granted or not to perform some specific actions (gold customer, business partners, etc..)

For internet access, the following requirements are mandatory

- Confidentiality: sensitive data must be encrypted (credit card payments)
- Data integrity : Data sent across the network cannot be modified by a tier
- Auditing: Every sensitive action can be logged
- Non-repudiation : gives evidence a specific action occurred

J2EE security model will be reused

## 3.4 Persistence

Data persistence will be addressed using a relational database.

## 3.5 Reliability/Availability (failover)

The availability of the system is a key requirement by nature, as it is a selling system. The candidate architecture must ensure failover capabilities. Reliability/Availability will be addressed through the J2EE platform

Targeted availability is 16/7: 16 hours a day, 7 days a week

The time left (8 hours) is reserved for any maintenance activities

### 3.6 Performance

The payment process (credit card authorization and confirmation) must be under 10 seconds.

### 3.7 Internationalization (i18n)

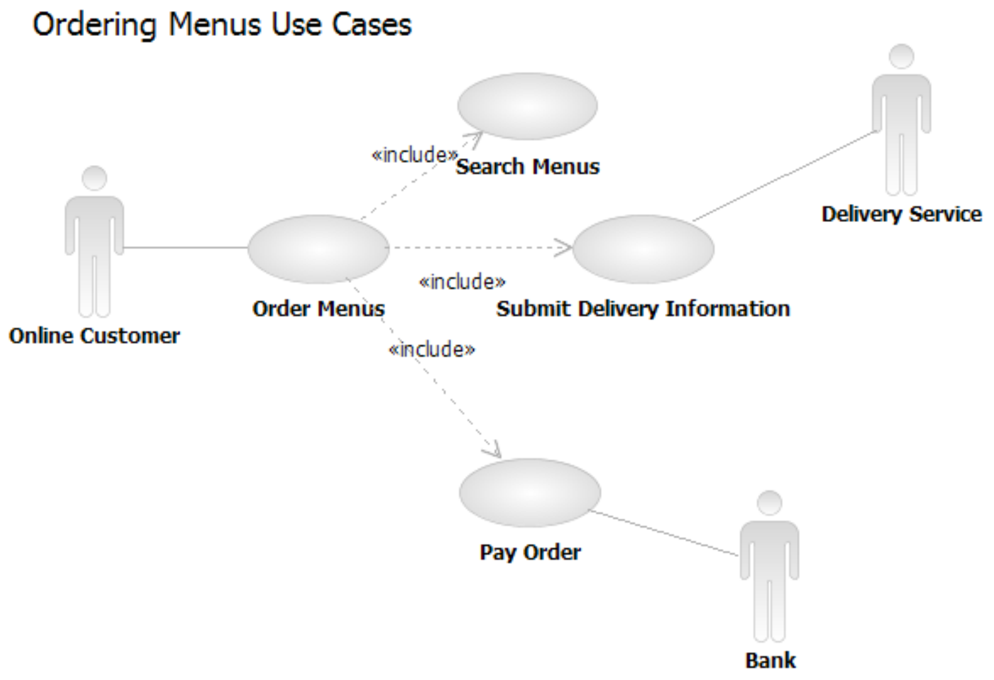
The online catering service of Yummy Inc must be able to deal with several languages (at least French and English)
So the presentation layer must be able to support i18n.
Other layers must be generic enough to work with any internationalization context

## 4. Use-Case View

This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system. The only use-case with a significant impact on the online catering architecture is the one related to online orders. It includes a search feature as well as a call to external services (delivery and payment)

### 4.1 Ordering Menus

A customer accesses the online catering application and search for the available menus. The customer chooses from a list of menus and select what she/he wants to order. Then, the customer performs an online payment (credit card). Once the payment has been validated, the customer confirms the order, enters her/his delivery information (name, address, phone number, etc..) and all the relevant information is sent to the Yummy Inc delivery service.



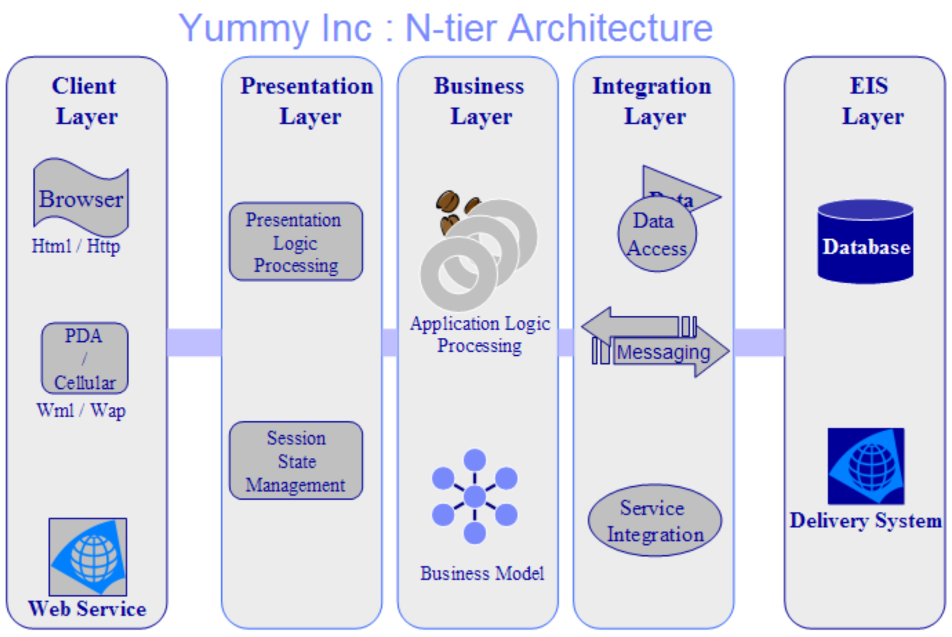
### 4.2 Use-Case Realizations

Refers to section 5.2 to see how design elements provide the functionalities identified in the significant use-cases.

## 5. Logical View

### 5.1 Overview

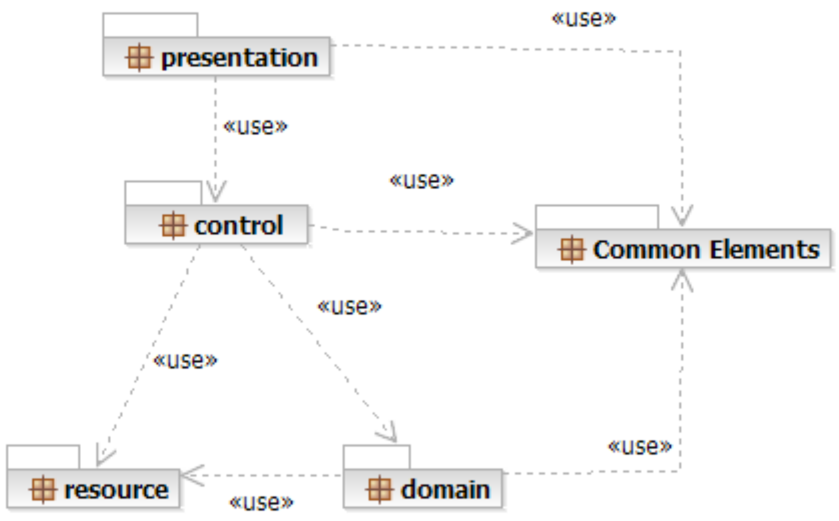
The online catering application is divided into layers based on the N-tier architecture



The layering model of the online catering application is based on a responsibility layering strategy that associates each layer with a particular responsibility.

This strategy has been chosen because it isolates various system responsibilities from one another, so that it improves both system development and maintenance.

## Architectural Layer Dependencies

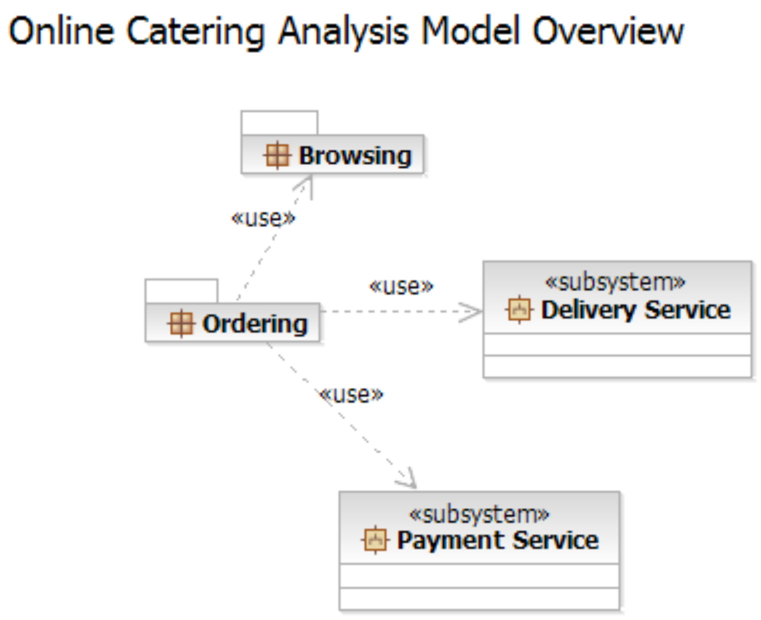


Each layer has specific responsibilities.

- The **presentation layer** deals with the presentation logic and the pages rendering
- The **control layer** manages the access to the domain layer
- The **resource layer** (integration layer) is responsible for the access to the enterprise information system (databases or other sources of information)
- The **domain layer** is related to the business logic and manages the accesses to the resource layer.
- The **Common Elements layer** gathers the common objects reused through all the layers

The online catering application version 1.0 is quite simple and only contains two basic features, one for the submit orders and the other allowing a customer to browse the online catalogue.

External services are reused fro the payment and the delivery functionalities.



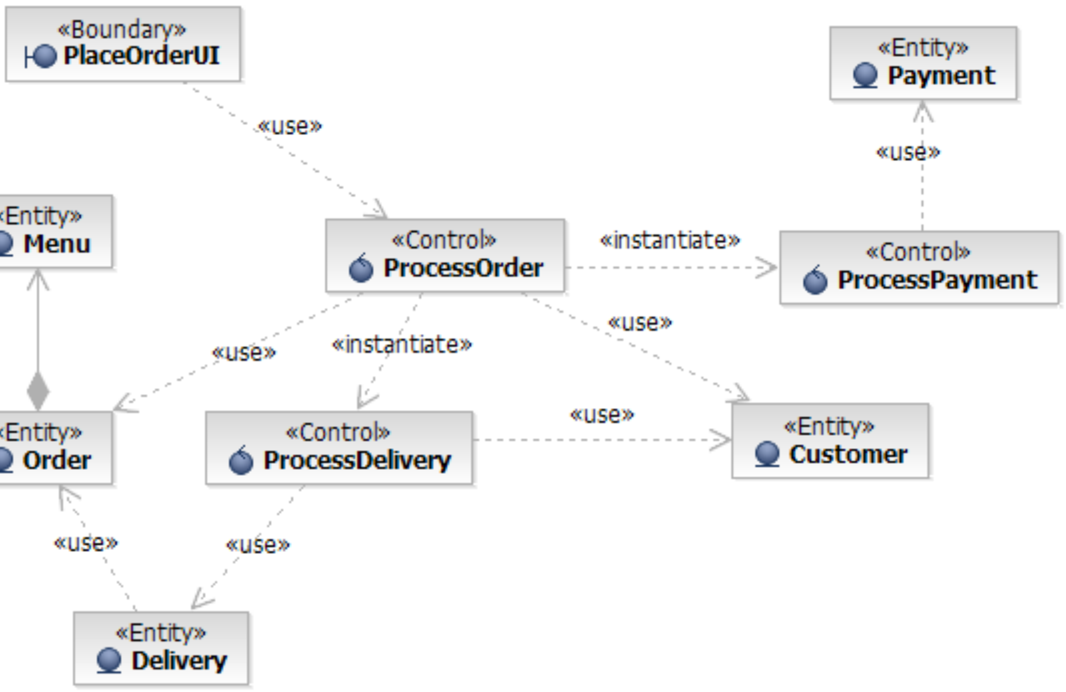
### 5.2 Architecturally Significant Design Packages

#### 5.2.1 Ordering Menus

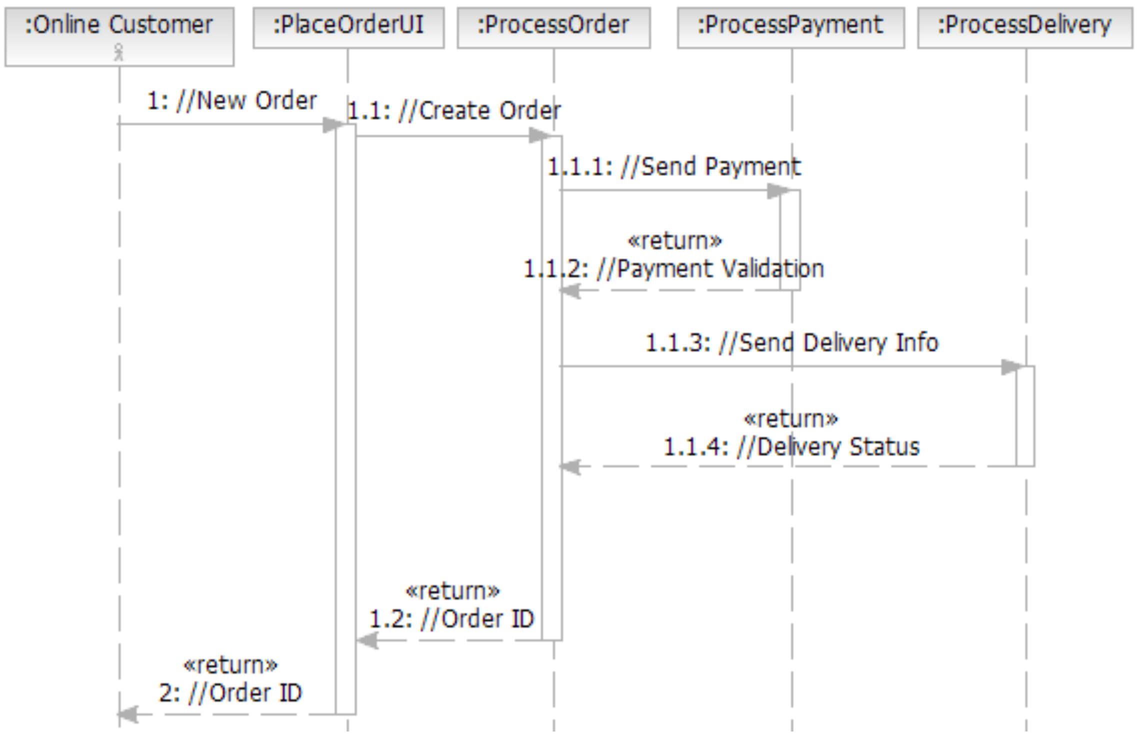
This package is responsible for all the logic related to the online orders. It provides ordering features and the necessary components to access the external services (Payment and Delivery)

| Analysis Model |
|---|

**Participants:**

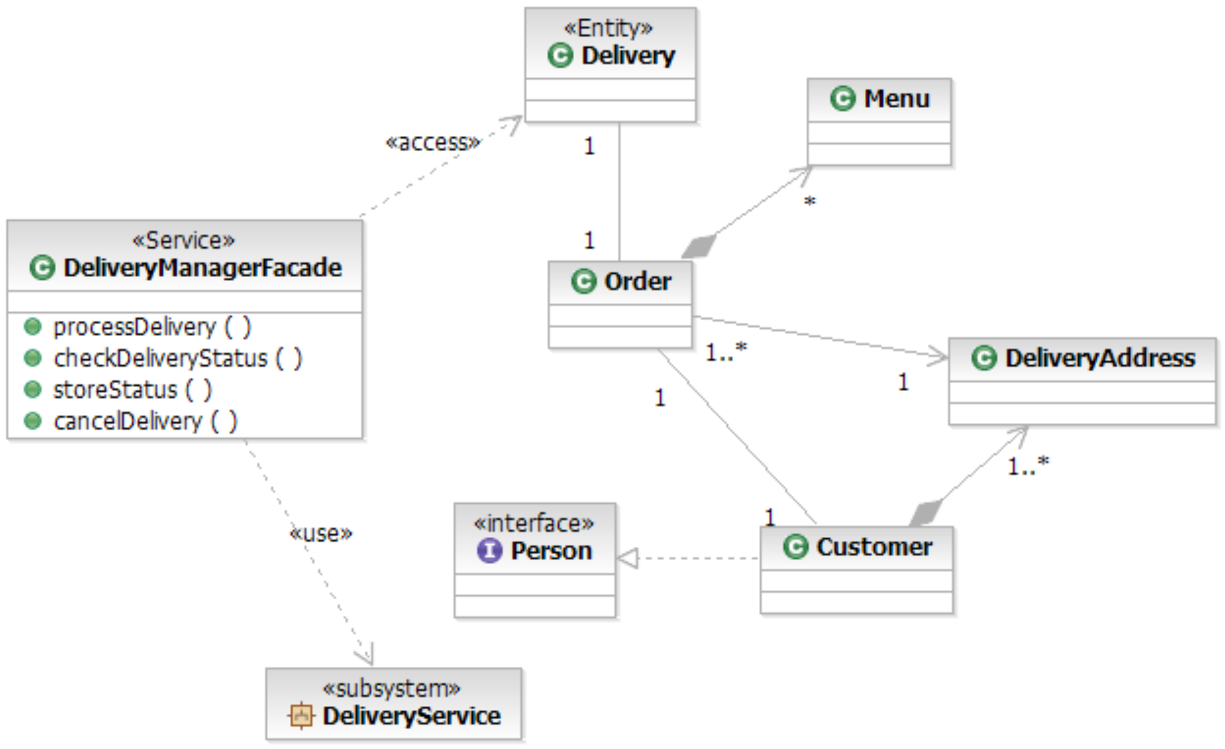


**Basic Flow:**



Design Model

**Process Delivery**



### 5.2.2 *Delivery Service*

Contains all the logic related to the Yummy Inc delivery service.
The Delivery Service is an external subsystem documented in its own Software Architecture Document

### 5.2.3 *Payment Service*

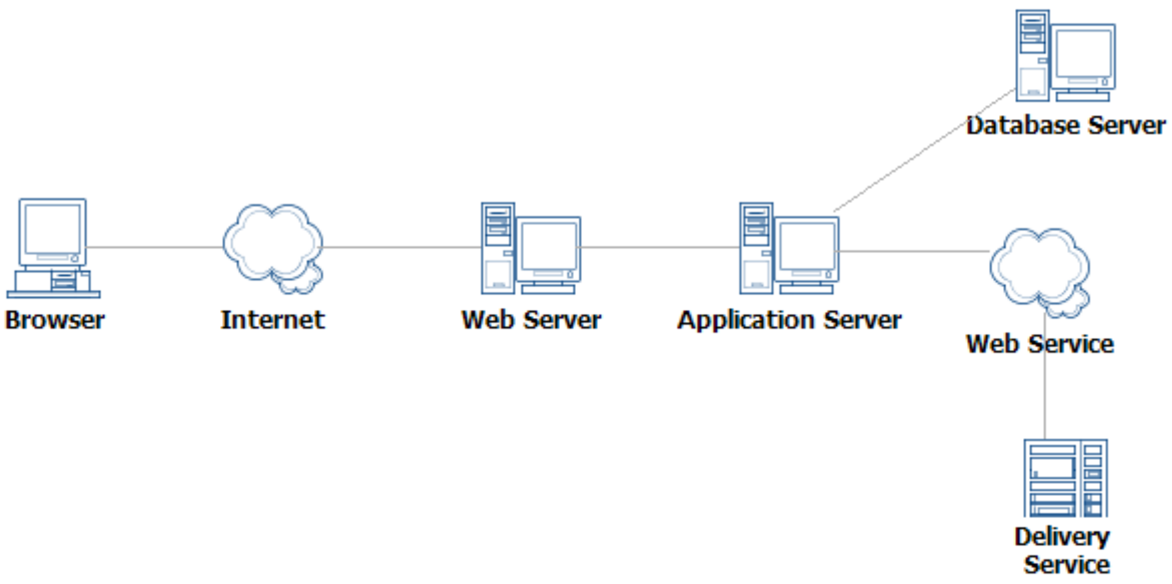
Contains all the logic related to the online payment and credit card validation.
The Payment Service is an external subsystem documented in its own Software Architecture Document.

# 6. Process View

There's only one process to take into account. The J2EE model automatically handles threads which are instances of this process.

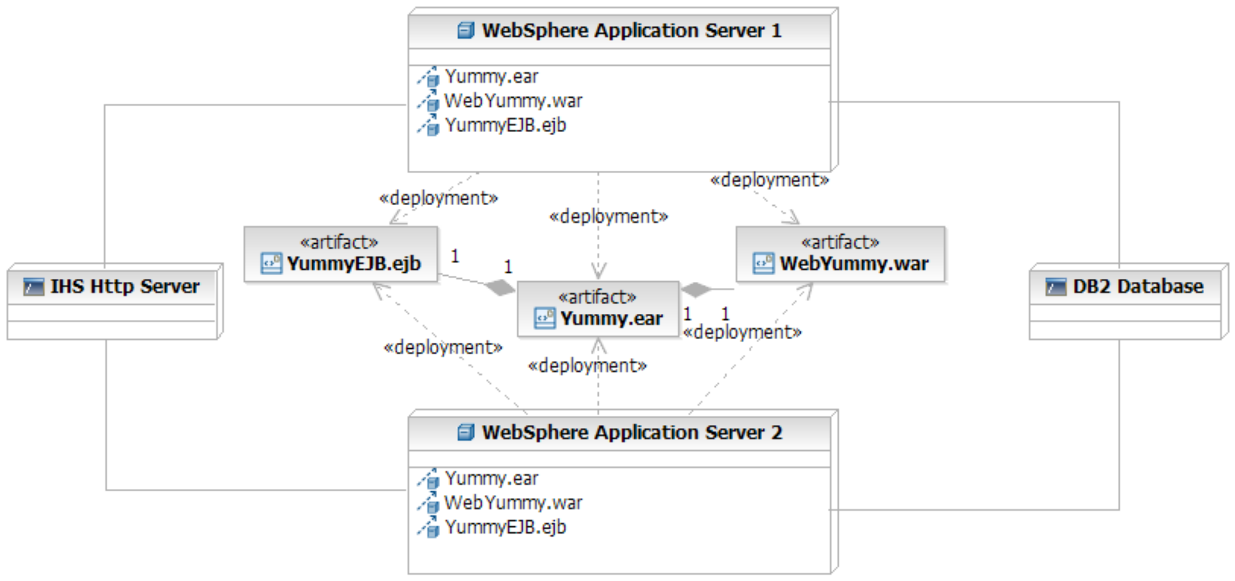
## 7. Deployment View

**Global Overview**



**Detailed deployment model with clustering**

- One IBM HTTP Server will dispatch requests to two different IBM WebSphere servers (load balancing + clustering)
- An IBM DB2 Database stores all the information related to online orders



# 8. Implementation View

## 8.1 Overview

The Implementation view depicts the physical composition of the implementation in terms of Implementation Subsystems, and Implementation Elements (directories and files, including source code, data, and executable files).
Usually, the layers of the Implementation view do fit the layering defined in the Logical view

It is unnecessary to document the Implementation view in great details in this document. For further information, refer to the Online Catering Service 1.0 workspace in Rational Software Architect.

## 8.2 Layers

### *8.2.1 Presentation Layer*

The Presentation layer contains all the components needed to allow interactions with an end-user. It encompasses the user interface

### *8.2.2 Control Layer*

The Control layer contains all the components used to access the domain layer or directly the resource layer when this is appropriate.

### *8.2.3 Resource Layer*

The Resource layer contains the components needed to enable communication between the business tier and the enterprise information systems (Database, external services, ERP, etc…)

### 8.2.4 Domain layer

The Domain layer contains all the components related to the business logic. It gathers all the subsystems that meet the needs of a particular business domain. It also contains the business object model.

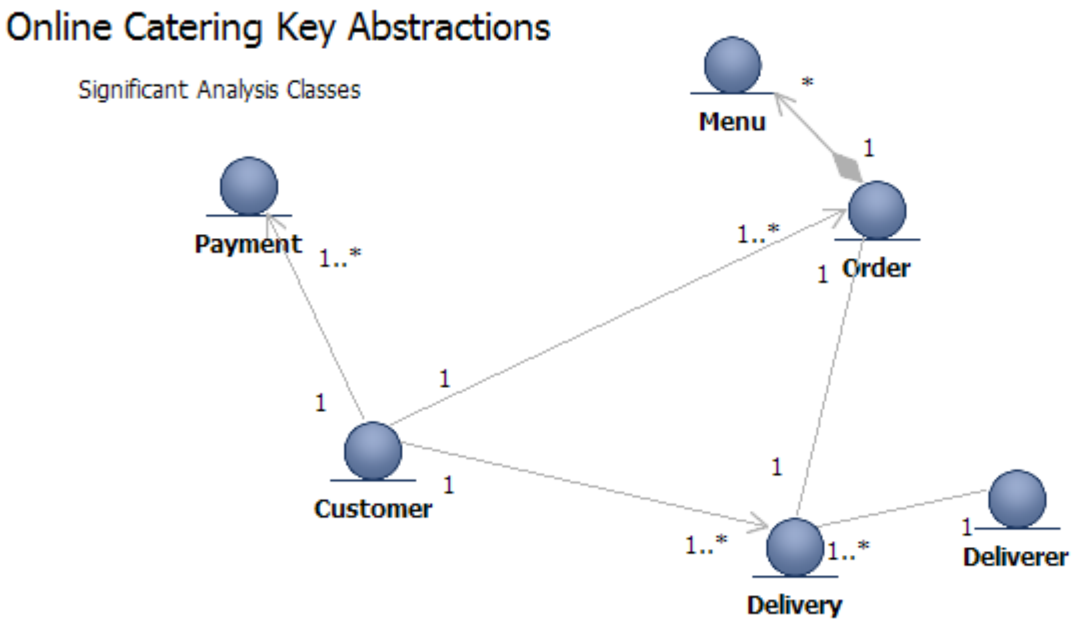
.

### 8.2.5 Common Elements Layer

The Common Element layer contains the components re-used within several layers.

## 9. Data View

The key data elements related to the online catering system are:



## 10. Size and Performance

Volumes:

- Estimated online orders : 100 a day, with peaks in the evening
- Yummy Inc registered individual customer : about 150
- Yummy Inc corporate customers : about 100

Performance:

- Time to process and online payment (credit card validation + confirmation) : less that 10 seconds required

## 11. Quality

As far as the online catering application is concerned, the following quality goals have been identified:

**Scalability**:

- **Description** : System's reaction when user demands increase
- **Solution** : J2EE application servers support several workload management techniques

**Reliability**, **Availability**:

- **Description** : Transparent failover mechanism, mean-time-between-failure
- **Solution :** : J2EE application server supports load balancing through clusters

**Portability**:

- **Description** : Ability to be reused in another environment
- **Solution :** The system me be fully J2EE compliant and thus can be deploy onto any J2EE application server

**Security**:

- **Description** : Authentication and authorization mechanisms
- **Solution :** J2EE native security mechanisms will be reused