Customer Complaint System

Bachelor Thesis
Version 1.0

Delft University of Technology
Bachelor Computer Science

Aryan Aras (1343475)
Charif Ennaciri (1180479)

Exam Committee
Ir. Cees Pronk
Ir. B.R. Sodoyer
Delft, July 8, 2009
# Table of Contents

Document status................................................................................................................. 3  
1 Summary ........................................................................................................................... 4  
2 Introduction ......................................................................................................................... 5  
   2.1 Assignment description ................................................................................................. 5  
   2.2 Company structure ....................................................................................................... 5  
3 Project Organization ........................................................................................................... 7  
   3.1 Project strategy ........................................................................................................... 7  
   3.2 Quality control ............................................................................................................ 8  
4 Analysis .............................................................................................................................. 9  
   4.1 Information gathering ................................................................................................. 9  
   4.2 Existing Systems ......................................................................................................... 9  
   4.3 Selected application ..................................................................................................... 15  
5 Research ............................................................................................................................. 16  
   5.1 Application .................................................................................................................. 16  
   5.2 Database ..................................................................................................................... 17  
6 Design ................................................................................................................................ 19  
   6.1 Design method ............................................................................................................. 19  
   6.2 Design decisions ......................................................................................................... 19  
7 Integration .......................................................................................................................... 22  
   7.1 DTAP .......................................................................................................................... 22  
   7.2 Hand over .................................................................................................................... 23  
8 Testing ................................................................................................................................. 24  
   8.1 Functional Testing ....................................................................................................... 24  
   8.2 Stress Testing .............................................................................................................. 25  
9 Evaluation ........................................................................................................................... 28  
   9.1 Experiences ................................................................................................................ 28  
   9.1 Suggestions for improvements ................................................................................... 28  
10 References ........................................................................................................................ 29  
A Glossary.............................................................................................................................. 30
Document status

Document title: BSc_Thesis_v1
Identification: BSc_Thesis_v1.pdf
Authors: Aryan Aras & Charif Ennaciri

Overview

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Status</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>11-07-2009</td>
<td>In Progress</td>
<td>Initial creation</td>
</tr>
<tr>
<td>0.2</td>
<td>12-07-2009</td>
<td>In Progress</td>
<td>Chapter 1, 2, 3</td>
</tr>
<tr>
<td>0.3</td>
<td>13-07-2009</td>
<td>In Progress</td>
<td>Chapter 4, 5, 6</td>
</tr>
<tr>
<td>0.4</td>
<td>14-07-2009</td>
<td>In Progress</td>
<td>Chapter 7, 8</td>
</tr>
<tr>
<td>0.5</td>
<td>15-07-2009</td>
<td>In Progress</td>
<td>Chapter 9, 10</td>
</tr>
<tr>
<td>1.0</td>
<td>16-07-2009</td>
<td>In Progress</td>
<td>Hand in mentor</td>
</tr>
</tbody>
</table>
1 Summary

Students of the Computer Science faculty at Delft University of Technology finish their Bachelor study by doing research in the form of a Bsc. Project. We as software technology students held our internship at Inter IKEA Systems B.V. in which we were responsible for finding and implementing a solution for the registration of customer complaints. This document is a summary of all the steps we took to come to a successful end solution.

We want to thank Cees Pronk and Bernard Sodoyer, professors at Delft University of Technology for supervising our team during this project. Furthermore, we want to thank Cheryl Gilbert, Matt Feigal, Annerieke Brauns, Hanna Westerberg and Jerry Moore of Inter IKEA Systems B.V. for their fantastic and pleasant cooperation and support.

Delft, July 15, 2009
2 Introduction

2.1 Assignment description

Nowadays every department store needs to have a method of recording any complaints received from customers in order to address that issue and keep a record for similar future cases; however, we recognized that Inter IKEA Systems B.V did not have a customer complaint application to monitor the complaints.

Additionally, there also was no database where the complaints could be saved and the history of these complaints could be tracked for future reference. That is why there was a big need for a client-based solution, which helps the company to record the complaints and keep an overview of them.

2.2 Company structure

Inter IKEA Systems B.V. is the owner and worldwide franchisor of the IKEA Concept. Their tasks is to franchise the IKEA Concept worldwide, ensuring that the IKEA concept is fully implemented by all IKEA franchisees, procuring for and transferring the knowledge of the IKEA way of doing things, as well as protecting and improving the IKEA Concept to maintain continual success.

Franchising of the IKEA Concept comprises two Business Units and two Staff Units: IKEA Concept, IKEA Concept Support, IKEA Markets and IKEA Concept Monitoring. [1] These departments are named ‘Business’ in this description. The contact person of The Business is Hanna Westerberg who acts on behalf of the departments mentioned before.

Inter IKEA Systems has it’s own IT Department which is responsible for producing, operating and maintaining the applications that are used by the company. Within the IT department, there is a subdivision called IT Products, which is responsible for developing most of the applications that are used by Inter IKEA Systems. IT Products is the contact point of the business and the IT department. The manager of IT Products is Cheryl Gilbert who is also the person who represents IT for The Business.
In our project, the functional requirements were formulated by the Business. This is because the need and request for the application comes from this department and the users of the application work for the Business. On the other hand, IT Products is the department which is responsible for the maintaining of the application. That is why it is very important to fulfill the technical requirements of IT Products. This relationship is showed in the figure above.
3 Project Organization

3.1 Project strategy

To come to a successful end solution a good preparation is inevitable. Especially when it involves a system that is used by different actors who use the same system. To prepare the implementation, we took in account the demands and wishes that are published in the RAD document. [2]

Since a project in practice is build by a team of specialists that have their own background and specialties, a good strategy is indisputable. The most suited developing strategy in our case for this project was Extreme Programming. ‘The methodology takes its name from the idea that the beneficial elements of traditional software engineering practices are taken to "extreme" levels, on the theory that if some is good, more is better.’ [3]

The development team consisted of two persons, which made it easy to communicate and decide small development decisions. Because Inter IKEA Systems have an open door policy, an appointment could be easily planned and the questions quickly be answered. The intensive communication with the end-user and the need to refine the requirements resulted into an adapted extreme programming method.

Practices:

- A pure form of extreme programming is programming in pairs, but this was not possible for us. In our case, every person will evaluate the other persons work.
- Every part of our reports and programming codes were evaluated before the deadline even if that task were in draft phase.
- There was multiple prototype testing with the users.
- We used the KISS (Keep It Simple Stupid) principle. That is why we wanted to implement a basic functional system.
- Our programming code and the comments had to be understandable for every developer.
3.2 Quality control

In this subchapter, we explain which steps we took to guard the quality, make the documentation, manage the application revisions, test the pilots and evaluating the result.

3.2.1 Quality

Throughout this period, we held several meetings with the Business to talk about our progress and design decisions. Every week we as a team held a progress meeting and half way of the project period, we held a midterm presentation.

3.2.2 Documentation

Below we list the documentation we developed for the different phases.

- **Requirements phase**: Orientation Report, Requirement Analysis Document
- **Design phase**: Architectural and Technical Design Document
- **Implementing phase**: List of changes
- **Verification**: Test plan
- **Maintenance**: Technical and user manual
- **Acceptation**: Final report

All the documentations were updated during the project since the functional requirements adjusted according to the result of user testing and evaluation with the Business.

3.2.3 Version control

For managing the project documentation and programming code, we used a remote repository for tracking the changes and maintain the saved data. There are numerous different free remote repositories available. We as project members are familiar with SVN (subversion control); so we used this system as a remote repository.

3.2.4 The pilots

During the project period, we held a few meetings with the Business in order to demonstrate the pilots. We also held a midterm presentation in attendance of the stakeholders. After the presentation, we updated the system according to the reactions we received during this presentation.
4 Analysis

4.1 Information gathering
Understanding the subject matter was a very significant part of the project as it helped us to reach the desired solution for the issue. That is why we arranged a meeting with Hanna Westerberg (Business contact person) and Cheryl Gilbert (IT contact person) separately as a first step to handle this problem.

Hanna Westerberg assisted our group by giving us a better insight of the functional requirements and the necessity of this application needed by Inter IKEA Systems. From our experiences, we knew it was very important to involve the users during the designing phase in order to develop a system that is accepted by the users. Active user involvement requires user’s deep commitment to the project. They assume responsibilities and participate on decision-making activities. One technique for active user involvement is to put the users managing requirements. [4]

In addition, Cheryl Gilbert helped us by clarifying the technical requirements for this assignment. One of the most important things that we got from Cheryl was the fact that the application had to be developed in Microsoft environments because Inter IKEA Systems uses this environment in most of its applications. This means that the application should be a web based application, which should be developed in C# / .NET (or something similar which people in house are able to use and take care of after the project ends).

4.2 Existing Systems
As is mentioned in the plan of approach [5], one of the goals of the project was due to the limited time and resources to choose an existing ticketing software system. We would redevelop this system as a project team according to the technical and functional requirements.

The technical requirements implicated that the application should be web based. It should also use Microsoft technology (ASP, .NET and C#). The database had to be using the Microsoft SQL Server.

The next phase existed of doing research about existing commercial/open source software in accordance with the requirements. To keep the scope limited we did research on three existing systems.
Jitbit Helpdesk v3

Jitbit Helpdesk was a very promising solution. The application is used in various domains (helpdesk environment) and applied in several major companies. The details of the software system are listed below. The application offered wide functionality and a good basis to adapt the application according to the Business requirements. Jitbit also granted permission to change the software according to a company’s needs.

<table>
<thead>
<tr>
<th>Jitbit Helpdesk v3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Developer</strong></td>
</tr>
<tr>
<td><strong>Website</strong></td>
</tr>
<tr>
<td><strong>Description from website</strong></td>
</tr>
<tr>
<td><strong>Price</strong></td>
</tr>
<tr>
<td><strong>Programming language</strong></td>
</tr>
<tr>
<td><strong>Functional Specifications</strong></td>
</tr>
<tr>
<td>Separated user roles/permissions</td>
</tr>
<tr>
<td>Friendly user interface</td>
</tr>
<tr>
<td>Possibility to add case categories</td>
</tr>
<tr>
<td>Possibility to automatically send e-mails</td>
</tr>
<tr>
<td><strong>Statistics/Reports</strong></td>
</tr>
<tr>
<td><strong>Advanced search engine</strong></td>
</tr>
<tr>
<td><strong>Knowledge base</strong></td>
</tr>
<tr>
<td><strong>Uniform Templates</strong></td>
</tr>
<tr>
<td><strong>Possibility to add attachments</strong></td>
</tr>
<tr>
<td><strong>Technical Specifications</strong></td>
</tr>
<tr>
<td><strong>Documentation</strong></td>
</tr>
<tr>
<td><strong>Programming code commented</strong></td>
</tr>
<tr>
<td>ID</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>
Slick Ticket

Slick Ticket is an open source software system which can freely be downloaded. The system is developed by one person (Stan Naspinski). Slick ticket is also used in a government agency with hundreds of users. After research we discovered that Slick did not fulfill the most of the functional requirements. [2] Which would mean that we had the option to skip some desired functionality or to search for an other existing system. The details of the system are listed below.

<table>
<thead>
<tr>
<th>Slick Ticket v1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Developer</strong></td>
</tr>
<tr>
<td><strong>Website</strong></td>
</tr>
<tr>
<td><strong>Description from website</strong></td>
</tr>
<tr>
<td><strong>Price</strong></td>
</tr>
<tr>
<td><strong>Programming language</strong></td>
</tr>
<tr>
<td><strong>Functional Specifications</strong></td>
</tr>
<tr>
<td><strong>Separated user roles/ permissions</strong></td>
</tr>
<tr>
<td><strong>Friendly user interface</strong></td>
</tr>
<tr>
<td><strong>Possibility to add case categories</strong></td>
</tr>
<tr>
<td><strong>Possibility to automatically send e-mails</strong></td>
</tr>
<tr>
<td><strong>Statistics/Reports</strong></td>
</tr>
<tr>
<td><strong>Advanced search engine</strong></td>
</tr>
<tr>
<td><strong>Knowledge base</strong></td>
</tr>
<tr>
<td><strong>Uniform Templates</strong></td>
</tr>
<tr>
<td><strong>Possibility to add attachments</strong></td>
</tr>
<tr>
<td><strong>Technical Specifications</strong></td>
</tr>
<tr>
<td><strong>Documentation</strong></td>
</tr>
<tr>
<td><strong>Programming code commented</strong></td>
</tr>
</tbody>
</table>
Liberum

Liberum is an open source helpdesk solution. The user interface is simple. The functionality is however very basic. Liberum is developed mainly by one person Doug Luxem. Liberum was implemented in several small companies. The source code lacks of documentation and the programming code is not commented. Also it uses old technology (ASP) which could lead to compatibility issues.

Liberum Help Desk v0.97

<table>
<thead>
<tr>
<th>Developer</th>
<th>Doug Luxem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Website</td>
<td><a href="http://www.liberum.org/">http://www.liberum.org/</a></td>
</tr>
<tr>
<td>Description from website</td>
<td>Liberum Help Desk is the complete help desk solution for small to medium sized businesses and organizations. This software provides a simple, easy to use web interface for managing and tracking technical support problems.</td>
</tr>
<tr>
<td>Price</td>
<td>Open Source</td>
</tr>
<tr>
<td>Programming language</td>
<td>ASP, VBscript and HTML</td>
</tr>
</tbody>
</table>

Functional Specifications

<table>
<thead>
<tr>
<th>Separated user roles/permissions</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Friendly user interface</td>
<td>Yes</td>
</tr>
<tr>
<td>Possibility to add case categories</td>
<td>Yes</td>
</tr>
<tr>
<td>Possibility to automatically send e-mails</td>
<td>No</td>
</tr>
<tr>
<td>Statistics/Reports</td>
<td>No</td>
</tr>
<tr>
<td>Advanced search engine</td>
<td>Yes</td>
</tr>
<tr>
<td>Knowledge base</td>
<td>No, it refers to a knowledge base but it exists of old closed cases.</td>
</tr>
<tr>
<td>Uniform Templates</td>
<td>No</td>
</tr>
<tr>
<td>Possibility to add attachments</td>
<td>No</td>
</tr>
</tbody>
</table>

Technical Specifications

| Documentation                  | No, not enclosed with the software |
| Programming code commented     | No, programming code is not commented |
Screenshot

Firefox

User Name: repdemo
Problems: 310

Support rep logged in.
Help Desk Administration

Demo

Help Desk
Submit New Problem
[View Problem List]

View problems for: repdemo观

Search Problems

Knowledge Base
Search the Knowledge Base

Other
Lookup by ID

Edit Information

User Menu | Rep Menu | Log Off

Return To The Librum Website

Librum Help Desk, Copyright (C) 2001 Doug Luxen. Please view the license.
4.3 Selected application

After we presented the three systems to both the IT Products department and the Business department, we had to choose a system we could use as a base. After short discussion, we all agreed that Jitbit Helpdesk was the most suitable application for this project. This decision was based on the following arguments.

- Jitbit Helpdesk offers wide functionality which means that after a few weeks of programming we could deliver an application that would suite most of the desired functionality.

- Jitbit Helpdesk uses ASP.NET technology, which is the same technology that is used in Inter IKEA Systems to develop other applications.

- A developers license is very affordable ($600,- which is about € 480,-)

- Jitbit also grants permission to change the application according to the company needs as long as the new application is not sold commercially.
5 Research

5.1 Application

After Jitbit Helpdesk was purchased, we had to do some research on the possibilities and architecture of ASP.NET and IIS. We did this by experimenting on a test server we had installed at home. Since we did not have any C# experience we had to become familiar with this popular programming language. We accomplished this with the use of online free tutorials.

Our goal was to make the design as simple as possible. The application should be fast to learn for new users, as this would lead to better acceptance of the intended users of the application. We therefore excluded and deleted the functionalities that would not be used by Inter IKEA.

ASP.NET uses a different kind of MVC (Model-View-Controller) design pattern than we were familiar with. Every web page consists of an aspx and an aspx.cs file. The aspx files stores X(HTML) markup and controls like buttons, checkboxes and dropdown list. The functionality of these controls and additional methods are defined in the corresponding aspx.cs file. Classes that are defined as the Businesslayer, which are also aspx.cs files, handle insertions and updates to the database. The extension CS stands for C# (referred as C Sharp). More details can be found in the class diagram, which is included with the documentation.

It was also very satisfying that Jitbit Helpdesk was designed in a way in which it made the application very re-usable. Additional functionality can be added by adding a new web page or by extending existing web pages without interfering with existing functionality since coupling is reduced. This can also be seen on the class diagram of our end solution (from now on referred as the customer complaint system).
5.2 Database

In the mean time on of our team members was responsible for doing research on the database. We found out that the Jitbit database was free of certain undesirable characteristics that could lead to a loss of data integrity since the database was well normalized.

Jitbit also used Microsoft SQL Server as a platform for running the database, which was ideal since Inter IKEA runs Microsoft SQL Server 2005 for their existing applications.

Since multiple users can log on to the system simultaneously, it was important to investigate what occurs when multiple users work on an issue at the same time. We can distinct these operations in two categories:

1. Read operations
2. Write operations

In the case of a read operation, this does not lead to problems since reading of information cannot lead to inconsistency.

In the case of a write operation, this is more complicated. One of the most important aspects of using a database is the reliability of the data. In order to guarantee the reliability the Microsoft SQL Server enforces data integrity. In our case this was useful when more than one user wants to update the same issue at the same time.

When a user executes an operation on a specific issue, the database management system will lock the row, which belongs to that issue and the relationships of that row. This means that the database works like a queue. When a query wants to perform an operation on a row that is locked, the operation is waiting in the queue until the specific row is released.

Another important aspect of the database is the fact that the database executes the operations based on time stamps (detailed in milliseconds).

In our case when more that one user wants to update an issue, the first coming update will be executed and the other will wait in the queue. In case that they arrive at the same time, one of them will be randomly selected and executed, in the mean time the other will be put in the queue.

The good thing about Microsoft SQL server is that from a technical standpoint, SQL Server does all of the work of implementing the ACID properties. The ACID properties consists the following important basic database principles. [8]
Atomicity - A transaction must be an atomic unit of work where either all of its modifications are made, or none of them are made.

Consistency - When finished a transaction must leave all data in a consistent state. In this case “consistent state” means that all rules, or constraints, are applied to a transaction’s modifications.

Isolation - Changes made by simultaneous transactions must be isolated from the changes made by any and all other simultaneous transactions. In other words, a transaction will either “see” the data in the state it was in before another simultaneous transaction modified it, or it will see the data after the second transaction has completed, but it will not see an intermediate state. This is known as serializability because it is possible to load the data back to a starting point, “replay” the transactions, and end up with data in the same state as it was after the original transactions were made.

Durability - Once a transaction is committed, the data modifications are permanent and can withstand system failure.

For more detail information we refer to the Architectural and Technical Design Document. [6]
6 Design

6.1 Design method

We used the existing Jitbit application as a base to build the new application that satisfies the needs of Inter IKEA Systems. A part of the Jitbit application fulfilled the functional requirements; however, we still had to adjust some of the implementation. Moreover, we had to implement new functionalities and change the style of Jitbit to the style of Inter IKEA Systems.

During our design we used the KISS (Keep It Simple Stupid) principle. [9] There were several involved parties in our project. The involved parties were the University, IT department of Inter IKEA Systems and the Business of Inter IKEA Systems. That is why the use of the KISS principle was very useful and helped us to break these involved parties down into separated independent parties. Furthermore, to break down the whole design and integration process into smaller processes and handle them separately.

In order to guarantee that we build an application which matches the style of Inter IKEA Systems, we involved a graphic designer of Inter IKEA to recommend us on the design of the application. In addition, the end users were involved during the designing of the user interface. The reason that we wanted the end users to be involved is the fact that they will be the ones who will use the application and we wanted to design a user interface that can be used by them without reading a manual first.

In the paragraph below, we explain some design decisions.

6.2 Design decisions

User Interface and Human Factors
The application will be used by two kinds of users, which are the main users and the administrators. One of the most important principles of Inter IKEA Systems is that fact that their applications look simple and basic. That was the reason that we decided not to use many colors and design a user interface, which will be simple to use directly.

Another very important design issue in our application is the fact that the user will get a specific error message in case the user executes an action, which is not allowed. A very simple example is when a user forgets to fill some fields. The application will give an error message and will indicate which fields have to be filled.
Hardware Considerations
Inter IKEA already had web- and database servers which they use for their existing applications. That is why there was no need to purchase a new server. We made use of the existing hardware, which they have.

As it is been explained in our RAD [2] an issue can include an attachment or more attachments in one issue. That is why we wanted enough disk capacity (at least 2 Gigabyte) to be reserved on the Database server. This was not a problem for Inter IKEA Systems.

Performance Characteristics
Performance is one of the most important measurements of a new application. Maximum five different users (main users and administrators) will use our application and there will be 15 issues in a month to register. To test if our application has an acceptable performance under these circumstances we executed a stress test. For the results of the stress test read chapter 8.2 about Stress Testing.

Error Handling and Extreme Conditions
As we indicated before there were two different kinds of users. An administrator has more rights in the application than a main user. In order to minimize the chance of an error, a main user sees only the functionalities, which he/she can execute in the application.

Another major point is the fact that our application gives specific error messages in case a user tries to execute an invalid action. The most important examples of error handling and extreme conditions are listed below.

Example 1: New Issue
In case you forget to fill the mandatory fields, the application gives an error message for every field.

Example 2: My profile
In case you type two different passwords, the application gives an error message “Passwords do not match!” When you type two passwords, which match but are shorter than six characters, the application gives an error message “Password must be at least 6 characters long”.

Example 3: My Issues
In case you close an issue, the message log part will be locked and you cannot enter any update any more.
Quality Issues

Quality of the application was also a major point, which we took seriously. The application should be reliable. Since the application will run on a Microsoft IIS web server and a Microsoft SQL database server of Inter IKEA Systems, they will guarantee the reliability.

The application is used during office hours and the application does not perform any kind of scheduled operations after office hours. This means that the application can be brought down (outside office hours) by the technical administrators to perform maintenance on the web- and database server.
7 Integration

7.1 DTAP

One of the technical requirements of IT was to adapt the DTAP principle (Development → Test → Acceptance → Production). The IT department of Inter IKEA uses this principle. This helps developers to develop an application and test it separately without influencing the operation of other applications.

As a developer, you follow the steps and when you pass the first three steps, you can put the application on production where it can be used by the users.

Figure 2 DTAP principle
The DTAP principle consists of the following phases

**Development:** This is the environment where we adjusted and implemented the application. This environment was totally separated from the test and production environment.

**Test:** When we finished the implementation of the functionalities, we installed the application on a clean server, which is referred as the test server. On this server we executed all the test cases in order to test all the functionalities. Furthermore, we executed a stress test on this server in order to test the performance and guarantee a reliable application.

**Acceptance:** When the application was tested on the test server and passed all the test cases, we installed the application on the acceptance server. Then the test cases were executed again, but this time by the users. This was also the environment where the user gave their approval of the application.

**Production:** After the acceptance we sent our application to the production. This environment is separated from the development environment and the final environment in which the application is used.

### 7.2 Hand over

After the project there has to be a party, which is responsible for the support of the application. IT Helpdesk is a subdivision of the IT department of Inter IKEA. IT Helpdesk provides the first line support to users of Inter IKEA. Therefore, we had to follow a procedure to hand over our application.

One of the requirements of the hand over was to write support documentation. That is the reason we had to write a document in which we give solutions to most common questions or problems that we predict for the future. We also signed an agreement with IT Helpdesk.
8 Testing

8.1 Functional Testing

As explained before we used Extreme Programming. Every time when we implemented a part of the application, we tested that completely. In order to test every functionality of the application we made a Test plan document. [7] In this document, we wrote a test case for each use case. Moreover, we wrote a test case of every invalid action that can be executed by a user. A very simple and important example is when a user tries to login with wrong credentials and the application gives an error message of “Wrong credentials!”

After the development process, we executed every test case of the Test plan Document on the test environment. In addition, every test case is executed again on the acceptance environment by the users.
8.2 Stress Testing

Performance is an important point, which has to be taken in account during the development. In order to test the performance and reliability of our application we performed a stress test. We expanded the database content with a script that recorded our browser activity. For this, we used the Microsoft Web Application Stress Tool, which is an optional component of the IIS web server.

Figure 1 Screenshot Microsoft WAS tool
We built a script that performed the following steps:
1. Load the main page. It should be taken in account that for this operation the client makes multiple connections with the server to download images, buttons, javascripts and the HTML content.
2. Login with a correct username and password.
3. Connect to the database to check if the credentials are correct
4. Load the dashboard page.
5. Go to the new issue screen
6. Fill the fields of a new issue
7. Submit an issue
8. Connect to the database and insert the issue
9. Log out

The script executed simultaneously in different threads to simulate multiple users inserting new issues at the same time. We tested that in different scenarios.

**Scenario 1:**
Number of concurrent users: 2
Duration: 30 seconds
Result: 65 issues inserted in the database

**Scenario 2:**
Number of concurrent users: 4
Duration: 30 seconds
Result: 102 issues inserted in the database

**Scenario 2:**
Number of concurrent users: 6
Duration: 30 seconds
Result: 124 issues inserted in the database
We can conclude that in case the number of concurrent users increases, the number of inserted issues also increases. However, the ratio (number of concurrent users/new issues inserted in the database) decreases.
9 Evaluation

9.1 Experiences

We as a project team are very pleased to have had the opportunity to perform a project for a major international company as Inter IKEA Systems. We took this assignment very serious and we are pleased with the result.

We improved our skills involving the communication with different stakeholders. We did our best to please every stakeholder. However, we knew that sometimes, as future software engineers, we had to keep the scope limited in order to deliver a well-produced application.

It was also very satisfying that we could do a software development project from the start until the end (from requirements gathering, implementing, testing, to maintenance). We thereby could think of all kinds of aspects that are important in order to produce a system that is accepted by the users, well documented and simple to use.

We also have learned how to adapt the DTAP principle, which is a very important principle that is widely used in information systems. Moreover, we got a better view how projects are performed in the industry. Shortly, this has been the most valuable experience during our study.

9.1 Suggestions for improvements

As a software engineer you have to be honest about your developed system and always criticize the result. In our case we have suggestions for improvements on the following aspects:

- The use of Active Directory, this means that we do not have to use a separate user account administration. The application can be extended in order to have a single sign on feature based on the windows account of the user (which also relies on Active Directory).
10 References

[1] Inter IKEA, Facts about the IKEA business, 2009
Intranet Inter IKEA


[8] Mike Aubert, SQL Server Tables and Transactions, 2003

http://nl.wikipedia.org/wiki/KISS-principe
A  Glossary

ASP.NET
a web application framework developed and marketed by Microsoft to allow programmers to build dynamic web sites, web applications and web services.

ATDD
Architectural and Technical Design Document

C#
(pronounced C Sharp) is a multi-paradigm programming language that encompasses functional, imperative, generic, object-oriented (class-based), and component oriented programming disciplines. It was developed by Microsoft as part of the .NET initiative.

CCS
Customer Complaint System

IIS
formerly called Internet Information Server - is a set of Internet based services for servers created by Microsoft for use with Microsoft Windows.

Issue
each complaint that is registered in the application is called an issue

Knowledge Base
a portal where existing issues can be added for future reference. This in order to share handy solutions between the main users and administrators.

RAD
Requirements Analysis Document

Report
graphical representation of statistics

SQL
Structured Query Language

Template
pre defined e-mail text templates in order to send uniform e-mails to customers

Use case
a use case in software engineering and systems engineering is a description of a systems behavior as it responds to a request that originates from outside of that system. In other words, a use case describes "who" can do "what" with the system in question.