Preface

The third year of the study Computer Science at the Delft University of Technology (DUT) is completed by an internship. This internship can take place either in The Netherlands, or in a foreign country the DUT has connections with. We chose to go for the latter option. The report you're currently reading is the result of our two month's internship at the University of San Carlos (USC), located on the island of Cebu in the Philippines.

This report assumes some knowledge in the field of Software Engineering. For reasons of brevity and clarity concepts like UML or ITIL are not elaborately discussed. [Bruegge & Dutoit, 2000] and [OGC, 2000] can be consulted for more information on these topics.

To maintain readability the authors wrote 'he' and 'his' at places where 'he/she' and 'his/her' were meant. For Filipino's, this difference is very vague anyway…

We would like to thank the following people that have contributed to the success of this project.
From the Netherlands: Bernard Sodoyer and Bert Geers, who gave us the opportunity to go here in the first place. Gerdien de Graaf, Jan de Vries and Rien Dam, who assisted us in making this trip possible financially and gave us very valuable advice.

Everybody at IRM: Bennie Revecho, Bianca Vallejos, Daryl Daño, Garry Patigayon, Glen Bacalla, Jan Amistad, Jayson Saraosos, Jessie Dalde, Joeffrey Aquino, Karl Bajenting, Leo Chiang, Mario Cúñado, Mejín Rubio, Michelle Cansana, Nola Akut, Rhea Rivera, Rodill Castillon, Tito Pijano and Winston Cordova, for the nice working climate, the lunches together and great nights out.

The other international students who also stayed at Girls High: Bas and Wietse, for showing us around in Cebu, Anne and Christel, for the great diving course together and Saskia, for continuing our efforts by coordinating the further implementation of BORREL into the IRM organization.

And of course we'd like to thank all Filipino's we haven't mentioned before, for their friendliness, and especially all Filipina's, for being so incredibly pretty!

Arjen Kruithof, Jelle ten Hoeve, Leendert Breukel
Cebu City, september 2003
Table of contents

Preface 1

Table of contents 2

1 Management summary 4

2 Definitions & abbreviations 5

2.1 Definitions 5

2.2 Abbreviations 5

3 Introduction 7

3.1 The background 7

3.2 The objective 7

4 Requirements analysis 9

4.1 Introduction 9

4.1.1 Purpose of the system 9

4.1.2 Scope of the system 9

4.1.3 Success criteria of the system 9

4.2 Current system 9

4.3 Proposed System 11

4.3.1 Overview 11

4.3.2 Functional Requirements 11

4.4 Nonfunctional Requirements 11

4.4.1 User Interface & Human Factors 11

4.4.2 Documentation 11

4.4.3 Hardware Considerations 11

4.4.4 Performance Characteristics 11

4.4.5 Error handling and extreme conditions 11

4.4.6 System modifications 11

4.4.7 Security issues 11

4.5 Pseudo requirements 11

4.6 System models 11

4.6.1 Use case model 11

4.6.2 Object Descriptions 11

4.6.3 Class diagram 11

4.6.4 Dynamic models - State Charts 11

4.6.5 User interface Navigational paths & mock-ups 11

5 System Design 11

5.1 Introduction 11

5.2 Design Goals 11

5.2.1 Performance 11

5.2.2 Dependability 11

5.2.3 Cost 11

5.2.4 Maintenance 11

5.2.5 End user 11

5.2.6 Trade-offs 11
5.3 Current software architectures for similar systems
  5.3.1 Software related to similar projects
  5.3.2 The rest of the playing field

5.4 Proposed software architecture
  5.4.1 Design considerations
  5.4.2 Subsystem decomposition
  5.4.3 Hardware/software mapping
  5.4.4 Persistent data management
  5.4.5 Access control and security
  5.4.6 Global software control
  5.4.7 Boundary conditions

5.5 Subsystem services
  5.5.1 Acquisition
  5.5.2 Configuration
  5.5.3 GUI
  5.5.4 Incident
  5.5.5 Personnel
  5.5.6 Report

6 Implementation
  6.1 Synchronization and notification

7 Conclusions
  7.1 RAD and SDD
  7.2 Implementation
  7.3 Passing on BORREL to IRM-ACS

8 Recommendations
  8.1 General recommendations
  8.2 Specific development recommendations

References

Appendix A - Requirements analysis

1 Current System
  1.1 Flow Charts

2 Proposed System
  2.1 Use Case Model

Appendix B – System Design

1 Two Tier vs. Three Tier
  1.1 Two tier architectures
  1.2 Three tier architectures
1 Management summary

The BORREL Helpdesk Tool project aims to improve adherence to Service Level Agreements (SLA’s) by the Information Resource Management (IRM) staff of San Carlos University, and to provide management with better insight in the department.

The above goal was met by designing a comprehensive software tool that tracks all enquiries to the 1st line support up to the moment all issues have resolved. Not only does this supply the support staff with the means to assess the current situation in a moments time, but it also provides the management with valuable information with which to leverage the departments efficiency.

BORREL relies on the powerful Microsoft SQL Server Database Management software as a centralized data-store. Application logic for the IRM staff is implemented in a Java-client, while a PHP based web application caters all other staff with progress reports on their incidents. BORREL. Industry standard practices combined in the IT Infrastructure Library (ITIL) were incorporated into the system, resulting in support for Incident Management and the possibility to extend to a full configuration management database (CMDB).

With the future in mind main focus was on system design, while ensuring the implementation phase would deliver a useable system. This way, the next development team already has the specifications for further implementation at their disposal. It is recommended that a policy is developed for integration of the new system into the organization. After that, more effort can be put into extending the implementation of the client application with the currently available system design as a foundation.
2 Definitions & abbreviations

2.1 Definitions

call
A physical phone call

Customer
The person who is producing a stimulus

Help Desk Employee
Someone who answers calls and updates log files (pull-out & helpdesk log) [Currently performed by Nola, Rhea, Winston]

Incident
Event which is not part of the standard operation of a service and which causes, or may cause, an interruption to, or a reduction in, the quality of that service.
NB: An Installation Request is also an Incident
NB2: An Incident is associated with a description of the Symptoms

installation
An installation of hardware-components, software-components or both

Problem
An Incident that cannot be handled over the phone (either by a Help Desk Employee or a Technician). In this case, it has to be handled by TNS or ACS.
This definition subtly differs from the ITIL-definition, by which a Problem is defined as 'The unknown underlying cause of one or more Incidents'.

site
The department where a stimulus originates and/or the problem is occurring.

stimulus
The input the helpdesk receives (Deprecated)

symptoms
The effect of an Incident that have been notified by the Customer

Technician
Someone who fixes stuff at IRM or on site

Unit
An object that is related to an Incident, like a mouse or a computer. This also includes parts like a motherboard.

2.2 Abbreviations

ACS
See IRM-ACS

ARIS
Academic Register Information System. This is the student registration system.

CI
Configuration Item

CMDB
Configuration Management DataBase

DBMS
DataBase Management System

DECL
DataExpress Component Library: The Data Access Framework offered by the JBuilder IDE

DUT
Delft University of Technology

FIS
Financial Information System
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOODS AL</td>
<td>Fully Object Oriented Database Synchronized Abstraction Layer</td>
</tr>
<tr>
<td>GNU</td>
<td>GNU is a recursive acronym for “GNU's Not Unix”</td>
</tr>
<tr>
<td>GPL</td>
<td>GNU General Public License: An Open Source license that allows free use for all purposes and editing of the code. [GNU, 2003]</td>
</tr>
<tr>
<td>HDIS</td>
<td>HelpDesk Information System</td>
</tr>
<tr>
<td>HDS</td>
<td>Help Desk Services</td>
</tr>
<tr>
<td>HIST</td>
<td>Helpdesk ITIL Support Tool</td>
</tr>
<tr>
<td>HRMIS</td>
<td>Human Resource Management Information System</td>
</tr>
<tr>
<td>IPDI</td>
<td>Information Policy Development and Implementation: The project in which the DUT and the USC cooperate to support the USC with its information systems</td>
</tr>
<tr>
<td>IRCM</td>
<td>Incident Registration and Configuration Management</td>
</tr>
<tr>
<td>IRM (1)</td>
<td>Information Resource Management</td>
</tr>
<tr>
<td>IRM (2)</td>
<td>Information Resource Manager</td>
</tr>
<tr>
<td>IRM-ACS</td>
<td>The department for Administrative Computing Service</td>
</tr>
<tr>
<td>IRM-TNS &amp; HDS</td>
<td>The department for Technical &amp; Networking Services and Help Desk Services</td>
</tr>
<tr>
<td>ITIL</td>
<td>Information Technology Infrastructure Library</td>
</tr>
<tr>
<td>LIBIS</td>
<td>Library Information System</td>
</tr>
<tr>
<td>OO</td>
<td>Object Oriented</td>
</tr>
<tr>
<td>OTS</td>
<td>Of The Shelf</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>TNS</td>
<td>Technical &amp; Networking Services</td>
</tr>
<tr>
<td>TNS &amp; HDS</td>
<td>See IRM-TNS &amp; HDS</td>
</tr>
<tr>
<td>USC</td>
<td>University of San Carlos: The university where the internship took place.</td>
</tr>
</tbody>
</table>
3 Introduction

3.1 The background

At one side of the world, situated at the island of Cebu and surrounded by palm trees, tropical heat and traffic chaos there is the majestic University of San Carlos. Beautiful Filipina's in school uniforms populate its interior. At the other side of the world, in the cold and rainy Netherlands, there is the Delft University of Technology (DUT) where we have spent the first 3 years of our studies in Computer Science. But one day these two worlds came together … We were given the opportunity to do our internship at the USC as part of the IPDI project, in which the TUD and the USC are participating in order to support the USC with its information systems. One part of this project is the establishment of an IT Help Desk, with a Help Desk Information System to support it.

When we arrived at the USC, the Help Desk was already functioning, as part of the Information Resource Management Department, but the administration was very rudimentary. Some basic facts were registered in a spreadsheet or on paper, while others were not registered at all.

As this Help Desk is servicing four campuses with a total of 17,000 students and IT is becoming more and more important for the functioning of the university, it is crucial that Service Level Agreements are met and customers can be adequately informed about the status of the incident they reported; even when the number of incoming calls is increasing.

3.2 The objective

Our job was to analyze the working of processes that are relevant to the Help Desk, optimize these processes if necessary and to design and implement an information system to support them. Furthermore, the staff that is going to use the system and the staff that is responsible in maintaining and expanding the software should be sufficiently trained.

As a method for developing the system we have chosen the incremental method. This means we first designed and implemented a full functioning, but basic version of the Help Desk Information System (HDIS), with only the must-have requirements met. Once the implementation of this first version (first increment) was finished and it was bug free to our knowledge, we installed the first version at the helpdesk and let the people work with it. When they discovered bugs or had any complaints about the program they could report to us. In the meantime, we were extending the first increment with a set of should-have requirements, making use of the feedback we got. And after the second increment is finished, the same procedure applies to the third one.
During the development we will explain the programmers how the HDIS is designed and guide them with the implementation of the other desired functions, which are not implemented in the basic version. In this way the system will evolve to the desired system and the users and programmers will be trained in using, maintaining and expanding the system.
4 Requirements analysis

4.1 Introduction

This chapter documents the Requirements Analysis Phase of the software project. The goal of this phase is to identify current procedures, and to elicit desired properties of the future system. These are used to model the future system entirely from the user’s perspective, using both informal (i.e. use cases) and formal (i.e. class diagrams) models.

4.1.1 Purpose of the system

The global goals of the HDIS are listed below.

1) Achieve the goals set by the SLA. These are mainly time limits.
2) Improve the possibilities to distill management information from the system, so organizational and executive adjustments can be made.
3) Increase the amount of calls the Helpdesk can handle per day with the same amount of resources.
4) Improve first line support, so more calls can be handled just by telephone.
5) Inform the customers by enabling them to track the status of their Problem through a website or even submitting an Incident via a website. This, too, will bring down the number of telephone calls to the Helpdesk.
6) Enrich the current specialization-process by suggesting an assignment of the most feasible technician to a certain problem.

4.1.2 Scope of the system

The system has to log all calls in a database, change the status of a call, print gate passes, show reports, etc. All interfacing with other systems, even the financial system, is out of scope. The human resource part is also beyond the scope of the system.

4.1.3 Success criteria of the system

To test if pre-set goals have been reached, measurable success criteria are defined. Although not all of the goals can be measured easily, we can derive success criteria for the following:

- Goal 1: Number of goals of the SLA that are reached.
- Goal 5: Number of calls to the Helpdesk regarding the status of a problem.
- Goal 4, 5: Percentage of number of calls that are handled by just first line support regarded to the total number of calls.

4.2 Current system
In this paragraph we describe the helpdesk processes as they are currently implemented. At the moment there is no specialized helpdesk information system present.

Before getting into the procedures a description will be given about the organizational structure of the helpdesk. Actually the IRM consists of two departments: The department for Administrative Computing Service (IRM-ACS) and the department for Technical & Networking Services and Help Desk Services (IRM-TNS & HDS).

IRM-ACS has five employees. The department developed the ARIS and maintains it. The Library Information System (LIBIS) is only maintained; it has been developed by an external company. In the future IRM-ACS will also maintain the Human Resource Management Information System (HRMIS) and the Financial Information System (FIS). Another company is developing these systems at the moment. The HDIS, which we are developing, will also be maintained by the IRM-ACS.

The IRM-TNS & HDS department has twelve employees. Due to reorganization last March the previous structure is still recognizable in this department. They have two helpdesk staff members, five technical staff members and four network staff members. The helpdesk staff primarily answers calls, network staff fixes and installs networks and the technical staff fixes and installs all other hardware and software related things.

In the whole process of Incident handling people of both departments can be involved. Besides the employee who is answering the phone, staff is needed from either IRM-ACS or from IRM-TNS & HDS for fixing problems or making acquisition recommendations. Last but not least, the head of IRM-TNS & HDS signs all kind of forms for approval and administrative purposes. These forms are listed below.

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Proposed</th>
<th>Objective</th>
<th>Signed by</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Requisition and Issue Slip</td>
<td></td>
<td>Obtain approval for the acquisition of hardware or a software license.</td>
<td>Staff member of requesting department, head of requesting department (Sometimes Rhea, Fr Jun)</td>
</tr>
<tr>
<td>2</td>
<td>Help Desk Information Log</td>
<td></td>
<td>To log all calls to the helpdesk</td>
<td>Nobody</td>
</tr>
<tr>
<td>3</td>
<td>Pull Outs</td>
<td></td>
<td>To log the pull-out of equipment from problem sites to the IRM-office.</td>
<td>Nobody</td>
</tr>
<tr>
<td></td>
<td>Form</td>
<td>Description</td>
<td>Issuer/Recipient</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Employee’s Gate Pass</td>
<td>To allow the technician to enter a certain building. One gatepass is required for a whole trip (Leave department A, enter B, leave B, enter A)</td>
<td>Employee, supervisor, guard (if going to another campus)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>(Equipment) Gate Pass</td>
<td>To allow the technician to bring stuff into a certain building. One gatepass is required for half a trip (Leave department A, enter B)</td>
<td>Issuer, Guard, Receiving department</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Call Ticket</td>
<td>V To inform the technician of the details of the job. (Not to enable the client to confirm whether the service (the solving of the problem) has been provided satisfactory)</td>
<td>Requesting department</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Pull out Form</td>
<td>V Just doubles the information already entered in the ‘Pull Outs’ sheet (3)</td>
<td>Nobody</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Installation Form</td>
<td>V To inform the technician of the details of the installation.(Not to enable the client to confirm whether the service has been provided satisfactory)</td>
<td>Requesting department</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Installations</td>
<td>To log all performed installations. Exact objective unknown.</td>
<td>Nobody</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Service-unit logbook</td>
<td>To log all service-units which are currently placed at some problem site</td>
<td>Nobody</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Network Recommendation Letter</td>
<td>To inform dept X of total cost of requested upgrades</td>
<td>Rhea</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Internet Application Form</td>
<td>V To apply for an internet connection</td>
<td>Applicant, Head, Dean, Technician, IRM</td>
<td></td>
</tr>
</tbody>
</table>

**Table 1: Forms used in the Current System**

Incoming calls are recorded on the Help Desk Information Log Sheet, which is implemented in MS Excel. On this sheet the status of a call is also administered. The call will be marked red when it is being handled and black when it has been closed.

An incoming call can be categorized in three categories:

1. A technical problem
2. An ACS software problem
3. An installation request

The different types of calls will be followed up by different processes. These are described in the appendix.

4.3 Proposed System

4.3.1 Overview

The proposed system supports the activities of the HDS, ACS and TNS (sub-) departments. Proposals for changing the current procedures are welcomed by the management. Requirements for this system have been divided into three groups: functional-, nonfunctional-, and pseudo requirements.

4.3.2 Functional Requirements

A functional requirement is an area of functionality the system must support. It describes interaction between actors and the system independent of the realization of the system. The requirements below are grouped by actor. The number in front of each requirement denotes its priority. A ‘(1)’ indicates the functionality will be implemented in the first increment, and so on. If the functionality of an entire actor has the same priority, the priority will be given next to this actor.

In this context, the term ‘modify’ means the modification any other attributes of an entity that are not mentioned in any of the other requirements.

4.3.2.1 Head of HDS & TNS

- (2) Approve all administrator actions
- (2) Approve recommendations
- (2) Approve the distribution of service units

4.3.2.2 Helpdesk Employee

- Stimuli (ACSS incidents, technical incidents and installation requests)
  - (1) Add, modify and delete stimuli
  - (1) Assign multiple technicians to a stimulus
  - (1) Relate configuration items to a stimulus
  - (3) Assign to technicians based on their skills
  - (1) Queue (when you cannot complete the form, for example when all technicians are busy or when de HE gets another call while filling out the form)
    - (1) Receive a notification when a SLA is nearly being breached

- Service units
  - (2) Lease and revoke lease of service units

- Printing
  - (2) Print pre-filled employee gate passes
  - (2) Print pre-filled equipment gate passes
4.3.2.3 Technician (3)

- Register, modify, release and delete pull outs
- Add, modify and delete configuration item types
- Add, modify and delete configuration items
- Search for precedents on base of problem description and/or configuration item type
- Create, modify, print and delete recommendations
- Modify stimuli

4.3.2.4 Manager

- Reports regarding stimuli (also pull outs, service units and installations)
  - (3) all stimuli
  - (3) stimuli per department
  - (3) stimuli per campus
  - (3) stimuli per attendant
  - (3) stimuli per day
  - (3) most appearing stimuli
  - (3) stimuli per support level
  - (3) most time consuming stimuli

- Other reports
  - (3) available technicians
  - (3) history per configuration item
  - (3) top ten of dumbest departments
  - (2) SLA compared to real situation (reactive)
  - (3) History of issued gate passes

4.3.2.5 Customer (2)

- (1) Track stimuli via internet
- (3) Submit stimuli via internet

4.4 Nonfunctional Requirements

Nonfunctional requirements pose user visible constraints on the system. They describe user visible aspects of the system that are not directly related with the functionality of the system.
4.4.1 User Interface & Human Factors
No constraints.

4.4.2 Documentation
For programmer reference a global documentation standard will be used, that is to be determined in preparation of the system design phase. This documentation will be used by the future maintainers of the system.

4.4.3 Hardware Considerations
The system preferably runs on the existing hardware- and network infrastructure. Current PC hardware is built around the x86 platform. Campus network communication is 2 Mbit, while intercampus connections to Main Campus are as follows:

- Talanban – 512kbit
- Boys high – 64kbit
- Girls high – 128kbit

4.4.4 Performance Characteristics
Those parts of the system that are going to be used for first line support must be highly responsive to provide quick feedback to the Customer. Two to ten concurrent users must be supported.

4.4.5 Error handling and extreme conditions
Any invalid user input should be noticed and some kind of feedback will have to be provided to the user prompting him to correct the input. System availability must be at least 95% during office hours.

4.4.6 System modifications
The system should be very flexible with regards to adding new functionality or changing business rules. Many additions can be imagined, and it should possible to implement them in the future.

4.4.7 Security issues
Because of the university-wide scope of some parts of the system, proper security is of great importance. Multiple access-right-sets should be provided to tightly fit the demands of each type of user. Users can dynamically be provided with one or more of these access-rights-sets. The access-right-sets itself, however, will be hard-coded. The head of IRM-HDS & -TNS will be able to perform user- & rights-management.

Basic measures should be taken to avoid malicious users from accessing the system through the network or the Internet.

4.5 Pseudo requirements
Pseudo requirements are constraints on the implementation of the system imposed by the client.

Since the IRM-ACS department will be handling the bulk of future modifications it is imperative that they are made familiar with the system starting in the early design phases. Where applicable, guidelines for design decisions can be provided by ACS staff to the development team. Software will be written in a language ACS staff is familiar with or will be made familiar with. Source code will be made available.

## 4.6 System models

### 4.6.1 Use case model

The Use Case Model acts as a basis for all other system models. It describes all tasks the system is to support, indicating their flow of events as interaction between users, forms and commands. Relations between use cases amongst each other and with all actors are depicted in the appendix.

Referred use cases are printed **bold**. Interface elements are in *italics*. A ‘form’ indicates editable information, whilst a ‘report’ is read-only.

### 4.6.1.1 General use cases

<table>
<thead>
<tr>
<th>Use case name</th>
<th>LogOn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating actors</td>
<td>Actor (initiator)</td>
</tr>
<tr>
<td>Entry condition</td>
<td>Actor wants to use the system</td>
</tr>
</tbody>
</table>
| Flow of events | 1. Actor starts the application if it is not already started  
2. If another user is currently logged in, but not using the system anymore, this user is logged out.  
3. Actor is presented with the *Logon Form* on which he chooses his name from the *Users List*  
4. Actor enters his password  
5. Actor submits the *Logon Form* |
| Exit condition | If login is correct, Actor is presented with the *Main Form* that’s personalized to the needs regarding the system. |
| Special requirements | The system is running on the *Workstation* |

<table>
<thead>
<tr>
<th>Use case name</th>
<th>LogOff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating actors</td>
<td>Actor (initiator)</td>
</tr>
<tr>
<td>Entry condition</td>
<td>Actor wants to stop using the system for now.</td>
</tr>
<tr>
<td>Flow of events</td>
<td>1. Actor just closes the application window, or uses the ‘<em>log off</em>’ command.</td>
</tr>
</tbody>
</table>
### Exit condition
User is now logged off the system through this particular Workstation.

### Special requirements
The system is running on the Workstation

#### SearchIncident

<table>
<thead>
<tr>
<th>Use case name</th>
<th>SearchIncident</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating actors</td>
<td>Actor (initiator)</td>
</tr>
<tr>
<td>Entry condition</td>
<td>Actor opens the ‘Search Incidents’ form</td>
</tr>
</tbody>
</table>
| Flow of events | 1. Actor is presented with some text fields in which he enters one or more search criteria. He can specify Customer name, Incident ID or a date range.  
2. Actor submits the search  
3. Actor is presented with a Similar Incidents List.  
4. Actor potentially selects several items to view their details. |

| Exit condition | Actor leaves the search incidents form |
| Special requirements | Actor is authenticated by the system. |

#### SearchPrecedent

<table>
<thead>
<tr>
<th>Use case name</th>
<th>SearchPrecedent (extends SearchIncident)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating actors</td>
<td>Actor (initiator)</td>
</tr>
<tr>
<td>Entry condition</td>
<td>Actor opens the search precedents form</td>
</tr>
</tbody>
</table>
| Flow of events | 1. Actor is presented with some text fields in which he enters search criteria such as keywords, equipment types or department name.  
2. Actor submits the search  
3. Actor is presented with a Similar Incidents List.  
4. Actor potentially selects several items to view their details. |

| Exit condition | Actor leaves the search precedent form |
| Special requirements | Actor is authenticated by the system. |

#### WriteGatePass

<table>
<thead>
<tr>
<th>Use case name</th>
<th>WriteGatePass</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating actors</td>
<td>Actor (initiator)</td>
</tr>
<tr>
<td>Entry condition</td>
<td>Actor needs to obtain an equipment or employee gate pass and is logged into the system</td>
</tr>
</tbody>
</table>
Flow of events
1. Actor chooses the Incident the gate pass will be related to from someone’s Assignments on the Main Screen or the ViewTechnicianAssignments use case.
2. Actor opens the Gate Passes Form from the Incident’s details.
3. A form displays which contains any previously created gate passes for the Incident, with some of their details. It also indicates whether these gate passes have been printed. The list contains Employee- as well as Equipment gate passes.
4. Actor can choose an item from the list, and its editable details will display elsewhere on the form. Actor can also opt to add a new gate pass.
5. Since the site’s department and therefore its campus is known, a lot of fields on the gate pass form will be pre-filled.
6. For an outgoing Equipment gate pass Actor will be able to select Units from three lists: available Service Units, New Units and Fixed Units. He selects one or more units from the list of which he knows that Technician wants to take them with him to the Site.
7. Actor chooses the ‘store’ command to enter the gate pass into the system.
8. The system asks whether the gate pass should be printed. The user can also select the ‘print’ command when viewing the details of a gate pass.
9. If the user chooses to print, the system will automatically create a standard gate pass report and tries to print it.
Exit condition
Technician can take the gate pass out of the printer for approval by Head. He can now go to the site.

Special requirements

4.6.1.2 Use cases for Helpdesk Employee

ViewTechnicianAssignments

<table>
<thead>
<tr>
<th>Use case name</th>
<th>ViewTechnicianAssignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating actors</td>
<td>Helpdesk Employee (initiator)</td>
</tr>
<tr>
<td>Entry condition</td>
<td>Helpdesk Employee wants to inspect what assignments a Technician is currently on.</td>
</tr>
<tr>
<td>Flow of events</td>
<td>1. Helpdesk Employee opens the Unfinished Assignments Form.</td>
</tr>
<tr>
<td></td>
<td>2. HE chooses a Technician from a Technicians List on who he wants to more information.</td>
</tr>
<tr>
<td></td>
<td>3. An Outstanding Assignments List is displayed. Their symptoms, creation date, helpdesk employee, and general status is displayed</td>
</tr>
</tbody>
</table>
## Exit condition
HE now knows what assignments the Technician has left.

## Special requirements
HE is authenticated by the system.

### NewInstallationRequestCall

<table>
<thead>
<tr>
<th>Use case name</th>
<th>NewInstallationRequestCall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating actors</td>
<td>Customer (stimulator)</td>
</tr>
<tr>
<td></td>
<td>Helpdesk Employee (initiator)</td>
</tr>
<tr>
<td>Entry condition</td>
<td>The Customer calls the Helpdesk because he wants to upgrade his network- or computer hardware.</td>
</tr>
<tr>
<td>Flow of events</td>
<td>1. Customer explains the reason of his call to HE.</td>
</tr>
<tr>
<td></td>
<td>2. HE opens the ‘New Call’ form of the HDIS, which contains several basic input fields to annotate the identity of the Customer and the issue he has brought up. The system automatically logs time information and the identity of HE.</td>
</tr>
<tr>
<td></td>
<td>3. HE uses the AssignTechnician use case to assign a Technician to the Installation Request. It does not matter whether the Technician is available right away because the process is not as time dependant as other processes. This technician will eventually inspect the site or make a recommendation right away.</td>
</tr>
<tr>
<td>Exit condition</td>
<td>If applicable, Customer is notified by HE of any actions that HDS &amp; TNS are taking to fulfill the request. The conversation ends, and HE clicks the ‘store’ button to permanently log the Incident.</td>
</tr>
<tr>
<td>Special requirements</td>
<td>Helpdesk Employee is authenticated by the system.</td>
</tr>
</tbody>
</table>

### NewIncidentCall

<table>
<thead>
<tr>
<th>Use case name</th>
<th>NewIncidentCall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating actors</td>
<td>Customer (stimulator)</td>
</tr>
<tr>
<td></td>
<td>Helpdesk Employee (initiator)</td>
</tr>
<tr>
<td>Entry condition</td>
<td>The Customer calls the Helpdesk because he is having difficulties with his computer.</td>
</tr>
</tbody>
</table>
Flow of events

1. Customer explains the reason of his call to HE.
2. HE opens the ‘New Call’ form of the HDIS, which contains several basic input fields to annotate the identity of the Customer and the issue he has brought up. At this point the HE also asks the user some relevant equipment specifications that HE looks up in a list of known Units, or adds to the system. The system automatically logs time information and the identity of HE.
3. HE Classifies the Incident according to the UnitType classification.
4. HE activates the SearchPrecedent use case.
5. The system returns a Similar Incidents List.
6. HE chooses one or more items from the list that appear similar to the issue at hand, and examines their details.
7. HE suggests some of the found solutions to Customer for a couple of minutes.
   a. If a solution has not yet been found, a Technician from the other room is called upon. If none is available, Customer is notified of this and should be queued for call-back at a later time. Use case ends.
   b. Otherwise HE clicks the ‘use this solution’ button for the appropriate Precedent, which copies the solution from the Precedent to the current Incident. HE can also type a custom solution. Use Case ends.
8. Technician talks the Incident over with Customer
   a. If a solution has not yet been found, HE assigns a (possibly different) Technician to the Incident to inspect it on-site with the AssignTechnician use case. If none is found, one has to be assigned at a later moment. Use Case ends.
   b. Otherwise Technician enters the given solution and steps to achieve it into the Incident report. This use case ends.

Exit condition

If applicable, Customer is notified by HE of any actions that HDS & TNS are taking to resolve remaining issues. HE also informs Customer of the assigned Incident ID for future reference.

The conversation ends. If the Incident does not need follow up actions, HE activates the HelpdeskCloseIncident use case. HE clicks the ‘store’ button to permanently log the Incident.

Special requirements

Helpdesk Employee is authenticated by the system.

QueryStatusCall

Use case name: QueryStatusCall
<table>
<thead>
<tr>
<th>Participating actors</th>
<th>Customer (stimulator)</th>
<th>Helpdesk Employee (initiator)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry condition</td>
<td>The Customer calls the Helpdesk.</td>
<td></td>
</tr>
</tbody>
</table>
| Flow of events       | 1. Customer explains the reason of his call to HE.  
|                      | 2. HE activates the SearchIncident use case.  
|                      | 3. HE informs Customer about progress on the Incident from the details provided by SearchIncident |
| Exit condition       | Customer is now informed about the progress on his Incident, conversation ends and HE closes all related forms. |
| Special requirements | Helpdesk Employee is authenticated by the system. |

**AssignTechnician**

<table>
<thead>
<tr>
<th>Use case name</th>
<th>AssignTechnician</th>
</tr>
</thead>
</table>
| Participating actors| Helpdesk Employee (initiator)  
|                     | Technician |
| Entry condition     | HE has to assign a Technician to a Incident |
| Flow of events      | 1. If HE already knows who has the time and the skills to attend the Incident, HE chooses this person from the Technician List. The use case ends.  
|                      | 2. HE opens the Technician Status report which tells him the whereabouts of all Technicians. The system automatically tries to sort the list based on Skills and Availability of Technicians. |
| Exit condition      | HE chooses a Technician and closes the report by assigning him to the Incident. |
| Special requirements| |

**CompleteIncompleteIncidents**

<table>
<thead>
<tr>
<th>Use case name</th>
<th>CompleteIncompleteIncidents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating actors</td>
<td>Helpdesk Employee (initiator)</td>
</tr>
<tr>
<td>Entry condition</td>
<td>HE wants to check if there are any incidents that have not yet been fully described, or which are still awaiting assignment to a Technician.</td>
</tr>
</tbody>
</table>
| Flow of events      | 1. HE looks at the Incomplete Incidents list on his main form, and chooses an entry he wishes to complete.  
|                      | 2. The New Incident Form appears  
|                      | 3. HE could start the AssignTechnician use case if a Technician has not yet been assigned, but should be.  
|                      | 4. HE could also fill out missing details in the input fields of the form.  
|                      | 5. If the Incident does not need follow up actions, HE activates the HelpdeskCloseIncident use case. HE clicks the ‘store’ button to permanently log the Incident. |
Exit condition  The Incident might be fully documented now, in which case it disappears from the *Incomplete Incidents list*.

**HelpdeskCloseIncident**

<table>
<thead>
<tr>
<th>Use case name</th>
<th>HelpdeskCloseIncident</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating actors</td>
<td>Helpdesk Employee (initiator)</td>
</tr>
<tr>
<td>Entry condition</td>
<td>HE has found a resolution for the incident over the phone (or Technician found it) and needs to tell the system about the details.</td>
</tr>
</tbody>
</table>
| Flow of events       | 1. HE chooses the ‘Close Incident’ Command  
                      2. HE is presented with a Close Incident Form on which he checks the logged solution for correctness and completeness. He also checks this for the ‘steps taken’.  
                      3. The Root Cause of the Incident is classified according to main- and sub classifications.  
                      4. HE submits the form |
| Exit condition       | The system now regards the Incident as being closed. |
| Special requirements | Helpdesk Employee is authenticated by the system. |

**NotifySLABreach**

<table>
<thead>
<tr>
<th>Use case name</th>
<th>NotifySLABreach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating actors</td>
<td>Helpdesk Employee</td>
</tr>
<tr>
<td>Entry condition</td>
<td>The system has found that an Incident has not been fixed within the time span the SLA indicates.</td>
</tr>
</tbody>
</table>
| Flow of events       | 1. Any one of currently logged in HEs will receive a notification on his screen alerting him of the breach.  
                      2. The HE in question acknowledges the notification |
| Exit condition       | HE could take real world actions to have this Incident prioritized. |
| Special requirements | Helpdesk Employee is authenticated by the system. |

### 4.6.1.3 Use cases for Technician

**ViewAssignments**

<table>
<thead>
<tr>
<th>Use case name</th>
<th>ViewAssignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating actors</td>
<td>Technician (initiator)</td>
</tr>
<tr>
<td>Entry condition</td>
<td>Technician has just finished a job, and wants to check which unfinished assignments are still waiting for him.</td>
</tr>
</tbody>
</table>
Flow of events

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.</td>
<td>If the <em>Unfinished Assignments list</em> is not already displaying, the Technician opens the <em>Unfinished Assignments Form</em>.</td>
</tr>
<tr>
<td>5.</td>
<td>An <em>Outstanding Assignments List</em> is displayed. Their symptoms, creation date, helpdesk employee, and general status is displayed.</td>
</tr>
</tbody>
</table>

**Exit condition**
The Technician now knows what assignments he has left.

**Special requirements**
The Technician is authenticated by the system.

---

**CheckInNewUnits**

**Use case name**
CheckInNewUnits

**Participating actors**
Head (initiator)

**Entry condition**
The retailer delivers new Units to IRM office.

**Flow of events**

1. Head opens the *Check-in New Units form* to log the arrival(s)
2. Head inspects the Units on completeness
3. Head opens the *Requested Units Form*
4. Head checkmarks all arrived Units on the *Requested Units Form*
5. Head submits the form

**Exit condition**
Head hands over the Units to the Technician assigned to the Incidents

**Special requirements**
Head is authenticated by the system.

---

**ReportJobFinished**

**Use case name**
ReportJobFinished

**Participating actors**
Technician (initiator)

**Entry condition**
Technician finishes (part of) a job

**Flow of events**

1. Depending on what type of job Technician has just performed he activates one or more of the following use cases:
   a. *CheckOutUnit*
   b. *CheckOutServiceUnit*
   c. *RegisterPullOutFixed*
   d. *RegisterPullOutIrreparable*
   e. *TechnicianCloseIncident*
   f. *WriteAcquisitionRecommendation*
2. Technician does *ViewAssignments*
3. Technician performs the *LogOff* use case

**Exit condition**
Technician now knows what to do next

**Special requirements**
Technician is authenticated by the system.

---

**CheckInUnits**

**Use case name**
CheckInUnits
**CheckOutUnits**

<table>
<thead>
<tr>
<th>Use case name</th>
<th>CheckOutUnits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating actors</td>
<td>Technician (initiator)</td>
</tr>
<tr>
<td>Entry condition</td>
<td>Technician wants to go to Site, and wants to tell the system he wants to take a repaired-, new- or service-Unit from IRM and install it at there.</td>
</tr>
</tbody>
</table>
| Flow of events    | 1. Technician chooses the Incident the Unit is related to from his Assignment list on the Main Screen.  
2. Technician opens the Check Out Unit Form from the Incident’s details  
3. He is presented with three lists: available Service Units, New Units and Fixed Units  
4. He selects one or more units from the list which he wants to take to Site.  
5. Technician submits the form.  
6. If applicable, the system shows all outstanding Service Units in the Incidents report. |
| Exit condition    | The system is notified that the Unit(s) will be taken from IRM |
| Special requirements | |

**CheckOutServiceUnits**

<table>
<thead>
<tr>
<th>Use case name</th>
<th>CheckOutServiceUnit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating actors</td>
<td>Technician (initiator)</td>
</tr>
<tr>
<td>Entry condition</td>
<td>Technician learned that the Incident can not be fixed within the time period mandated by the SLA or other parties, and decides a service unit should be issued to the Customer. He wants the system to suggest available Units, and to register the one he’s going to use.</td>
</tr>
</tbody>
</table>
| Flow of events | 1. Technician chooses the Incident the needed Unit is related to from his Assignment list on the Main Screen.  
2. Technician opens the Issue Available Service Unit form  
3. He uses some lists with brand names and Unit types to tailor the list of available units to include only the ones he’s interested in.  
4. Technician selects a suitable Service Unit  
5. Technician selects the ‘Issue’ Command to mark the Service Units as ‘in use’. |
| Exit condition | The system is notified of the Service Units that are going to be taken out of stock. There is no need to do CheckOutUnit now. |
| Special requirements | |

**TechnicianCloseIncident**

| Use case name | TechnicianCloseIncident |
| Participating actors | Technician (initiator) |
| Entry condition | Technician has resolved an Incident and needs to tell the system about the details. |

| Flow of events | 1. Technician chooses the Incident he wants to close from his Assignment list on the Main Screen.  
2. He chooses the ‘Close Incident’ Command  
3. Technician is presented with a Close Incident Form on which he checks the logged solution for correctness and completeness. He also checks this for the ‘steps taken’.  
4. The Root Cause of the Incident is classified according to main- and sub classifications.  
5. He submits the form  
6. The assignment will vanish from his Assignment List |
| Exit condition | The system now regards the Incident as being closed. |
| Special requirements | |

**RegisterPullOutFixed**

| Use case name | RegisterPullOutFixed |
| Participating actor | Technician (initiator) |
| Entry condition | Technician fixed a pull out Unit and wants to return it to the Customer. |

| Flow of events | 1. Technician tells HE that the pull out unit is fixed.  
2. HE opens the Pulled Out Units List. The rows show the description of the unit, the department it belongs to, the technician responsible for it, if a service unit is provided and perhaps more.  
3. HE selects the fixed unit by pointing it, than an Edit Pull Out Form shows. HE sets the state of the pull-out to fixed.  
4. HE tells the technician to re-install the Pull Out Unit and, if provided, to get the Service Unit back. |
Exit condition | Now the technician can physically re-install the Pull Out at the department.
--- | ---
Special requirements

### RegisterPullOutIrreparable

<table>
<thead>
<tr>
<th>Use case name</th>
<th>RegisterPullOutIrreparable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating actor</td>
<td>Technician (initiator)</td>
</tr>
<tr>
<td>Entry Condition</td>
<td>Technician concludes that a Pull Out Unit he tried to repair is irreparable and needs to take it out of rotation. He might also need a replacement Unit.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Flow of events</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Technician chooses the Incident the Dead Unit is related to from his Assignment list on the Main Screen.</td>
</tr>
<tr>
<td>2. Technician opens the Pulled Out Units List. He selects the Dead Unit by pointing it.</td>
</tr>
<tr>
<td>3. Technician uses the mark as irreparable command to mark the Unit as diseased.</td>
</tr>
<tr>
<td>4. Since the Unit is not useable anymore, the Technician might choose to issue a service unit using the CheckOutServiceUnit use case, and/or recommend the purchase of a replacement Unit by using WriteAcquisitionRecommendation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exit condition</th>
<th>The technician has taken steps to ensure (temporary) replacement of the broken Unit.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Special requirements</td>
<td></td>
</tr>
</tbody>
</table>

### WriteAcquisitionRecommendation

<table>
<thead>
<tr>
<th>Use case name</th>
<th>WriteAcquisitionRecommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating actors</td>
<td>Technician (initiator)</td>
</tr>
<tr>
<td>Entry condition</td>
<td>Technician has returned from Site, and needs to write a purchasing recommendation for the department Site is located at.</td>
</tr>
</tbody>
</table>
Flow of events

1. Technician chooses the Incident the recommendation will be related to from his Assignments on the Main Screen.
2. Technician opens the Write Recommendation Form from the Incident’s details.
3. He can now add entries to the list of recommended Units by selecting hardware brand and type from one or more product lists, and choosing an ‘add’ command.
4. Technician can also add custom entries in the provided text fields, and choosing the ‘add’ command.
5. Technician can also choose from a list of Available Units, so there’s no need to wait for the Units to be acquired.
6. Technician submits the form.
7. The system asks whether the recommendation should be printed.
8. If the user chooses to print the recommendation the system will create a standard recommendation letter and tries to print it.
9. The system adds the recommended Units to the list of requested Units if applicable, awaiting their arrival at IRM office.

Exit condition

Technician can take the recommendation out of the printer for approval by Head, and delivery to the requesting department.

Special requirements

4.6.1.4 Use cases for Head

PostponeRecommendation

<table>
<thead>
<tr>
<th>Use case name</th>
<th>PostponeRecommendation</th>
</tr>
</thead>
</table>
| Participating actors | Head (initiator)  
Customer (stimulator) |
| Entry condition | Customer calls Head to inform Head that the recommendation will not be followed (probably because of insufficient funds). The New Units associated with it will therefore not arrive at IRM office. |
Flow of events
1. Head opens the outstanding recommendations form
2. An Outstanding Recommendations List is displayed. Incident ID, recommendation date, Customer description and the recommended Units are displayed.
3. Head chooses the recommendation that will not be followed from the list, and selects the ‘Postpone’ command.
4. After confirmation by the user, the system removes the items associated with the recommendations from the List of Requested Units.
5. The system enters a description in the ‘solution’ field of the related Incident, indicating that the recommendation has been postponed indefinitely.

Exit condition The Incident can probably be closed now.

Special requirements Head is authenticated by the system.

4.6.1.5 Use cases for Manager

ViewReport
Use case name ViewReport
Participating actors Manager (initiator)
Entry condition Manager wishes to view management information regarding HD functioning.
Flow of events 1. On the Management Information Screen the manager selects the type of report to generate. (possible options are listed in this document under ‘functional requirements’)
2. Depending on the type of report, the system will present Manager with a Report Options Form that enables him to constrain the report conditions on date, employee name, department or campus.
3. Manager submits this form
4. Manager is presented with a form containing the requested report.
5. Manager views the information, and can optionally choose the print command to print the report onto paper.

Exit condition Manager is informed, and closes the report form.

Special requirements Manager completed the LogOn use case.

4.6.1.6 Use cases for Customer

QueryStatusThroughWWW
Use case name QueryStatusThroughWWW
Participating actors Customer (initiator)
Entry condition
Customer wishes to view status information on his Incident through the internet.

Flow of events
1. Customer starts his webbrowser
2. Customer opens the IRM-HDS website
3. Customer enters the Incident ID he obtained from a Helpdesk Employee at the time of reporting the incident.
4. Customer submits the form
5. Customer gets presented with the relevant details on his Incident. These include reporting time, his name, any actions a Technician has performed to resolve the Incident and what the general status of the incident is.
6. Customer closes his webbrowser

Exit condition
Customer is now informed.

Special requirements
Customer has a web browser and an active internet connection

**NewIncidentThroughWWW**

**Use case name**
NewIncidentThroughWWW

**Participating actors**
Customer (initiator)

**Entry condition**
Customer wishes to report an Incident through the internet.

**Flow of events**
1. Customer starts his webbrowser
2. Customer opens the IRM-HDS website
3. Customer browses through categories and subcategories of the FAQ.
4. Customer chooses a question or symptom resembling his own question resp symptom.
5. Customer gets presented with the relevant details on his Incident. These include reporting time, his name, any actions a Technician has performed to resolve the Incident and what the general status of the incident is.
6. Customer closes his webbrowser

**Exit condition**
Customer is now informed.

**Special requirements**
Customer has a web browser and an active internet connection

## 4.6.2 Object Descriptions

The application domain objects that have been identified during Requirements Analysis are documented below. Their property listings are not yet complete; they will be augmented during System Design.

### 4.6.2.1 Actors

Actors represent external entities that interact with the system; they can be human or other systems. The following actors have been identified with regards to the HDIS:

**User**
Any person who has permission to use the system. A User always has to identify himself to the system using authentication procedures.
**Head of IRM-HDS & -TNS (Head)**

*Position currently filled by: Rhea*

This actor is the supervisor of HDS- & TNS Staff. As such the Head will have to authorize many actions performed by this Staff. This includes gate-passes for equipment and personnel and purchase-recommendations.

**Helpdesk Employee (HE)**

*Position currently filled by: Nola, Winston, Rhea*

A Helpdesk Employee will answer support calls and provide first line support. This actor would also be the first to handle incoming email or faxes if such services would be provided to Customers. A HE will try to resolve Incidents from Customers. If necessary, control of the Incident will be transferred to a Technician. HEs remain the main contact regarding the Incident from the Customers point of view, although a Technician may also be assigned to an incident. HE can print gate passes for personnel and equipment.

**TNS & ACS Staff Member (Technician)**

*TNS Positions currently filled by: Ian, Glen, Rodill, Leo, Jessie, Rodel, Daryl, Bennie, Jayson.*

*ACS Positions currently filled by: Jeffrey, Karl, Bianca, Garry*

Technicians are called upon by HE’s to provide specialized help directly to the client, which means they provide second line support. They do this either by phone, or by on-site survey. The TNS Technicians transport failed hardware devices, service-units, software and new hardware between Sites and IRM. They make the more complex repairs at the IRM-TNS department, where they also assemble new hardware components. Whenever a Technician is called for an on-site survey due to an Installation Request he potentially creates a Recommendation letter to the Customer regarding hardware acquisition. ACS Technicians will deal only with problems related to the information systems they maintain.

As can be seen above, the two departments ACS en TNS have their staff generalized into the role Technician. But how do we tell who should attend an Incident? Since each Incident will be categorized, we know the type of an Incident when it becomes a Problem. Each type of incident will require certain skills from the person who attends to it. By storing the Skill-set of each Technician, each can have certain specializations, which make them the preferred attendant of certain types of problems.

**Finance Office**

The Finance Office decides if an acquisition-recommendation is within budget and will therefore be followed up by a purchase. It also takes care of accounting issues.

**Customer**

---

Internship Report • 2003-10-23, 12:27
Authors: Arjen Kruithof, Jelle ten Hoeve, Leendert Breukel
A Customer is also someone in charge of handling it at the side of the people having a problem, just like there is always a HE or Technician assigned to an Incident from within the IRM department. The customer calls the helpdesk and reports an Incident or asks for the status of an Incident. The customer can track Incident status through the internet.

Properties: Department

**Administrator**

Position currently filled by: Nobody

The administrator has control over the entire system, and is able to change anything he wants. This kind of access is required only sporadically (at most once a month)

**Manager**

Position currently filled by: Rhea, Fr Lagura

A manager needs access to most of the reports to obtain management information. He will not have to make any direct modifications to data contained in the system.

### 4.6.2.2 Other Entity Objects

Entity objects are objects which represent persistent or long lived information tracked by the system.

A note to the reader: the various kinds of Units have been transformed into one class Unit in the Class Diagram. This class has special properties to accommodate the perceived differentiation.

**Incident**

Event which is not part of the standard operation of a service and which causes, or may cause, an interruption to, or a reduction in, the quality of that service. Installation Requests are treated as Incident.

An Incident’s symptoms are the abnormalities that caused the Customer to report an Incident. It’s important to separate and keep track of the symptoms of a problem and a problem’s root cause. The cause of an Incident is identified by the Technician or the HE.

**Skill**

A competence a Technician can have, or that a certain Incident might need for resolution.

**Department**

A department at USC, located on any of the Campuses. This is where Customers are located.

**Site**

The place where an Incident is located; equivalent to Department.
Unit

A device, a part of a device, or a software license. A Unit can be a ServiceUnit, a PullOutUnit or a NewUnit. All are described below.

ServiceUnit

This special kind of Unit is owned by the IRM department, and can act as a temporary replacement for a Department’s broken Unit. It will be leased to the Department when the SLA time limit set to resolve an Incident won’t be met.

PullOutUnit

This special kind of Unit is taken from a Site by the Technician to be fixed at IRM office.

RecommendedUnit

This is a Unit that is not yet USC property, but has been recommended for purchase.

NewUnit

This special kind of Unit has never been used before. It’s generally ordered by a Customer and delivered at IRM. After delivery it needs to be installed at the Site. It could also be ordered in advance, so it is immediately available upon request.

Recommendation

A formal letter from Head to the Customer’s Department head with a list stating the exact hardware and software Units (incl. their cost) required to fulfill the Departments needs. It is created by a Technician, and approved by Head.

Assignment

Technician’s Assignments tell them what Incidents they currently have to resolve.

Service Level Agreement (SLA)

A document specifying what the responsibilities of IRM-HDS & -TNS are the Customer. For every category of Incidents there is usually a set time limit by which the Incident needs to be resolved.

SLA Paragraph

A SLA paragraph specifies allowed resolution time for a particular category of Incidents.

4.6.2.3 Boundary Objects
Boundary Objects are physical items or interface elements that enable the user to interact with the system. For most of these the use and the look will be fairly obvious, so we have only documented exceptional properties. Visual impressions for a selection of Boundary Objects can be found in the Screen Shots section of this document.

**Main Form**
For each employee function there will be a tailored main form which contains support for the most important tasks for the function.

**Workstation**
If Technicians are to be able to process their work by themselves, they will need separate workstations.

**Other Boundary Objects**
- Logon Form
  - Users List
- Search Incidents Form / Search Precedents Form
  - Similar Incidents List
- Gate Passes Form
  - List of Gate Passes
  - List of Available Service Units
  - List of Available New Units
  - List of Available Fixed Units
- Standard Gate Pass Report (Gate Pass)
- New Call Form
  - Incident Report
  - Technicians List
  - Technicians Status Report
  - Related Units List
- Incomplete Incidents List
- Close Incident Form
- Unfinished Assignments Form
- Check Out Unit Form
  - Technician Assignments List
  - Pulled Out Units List
- Edit Pull Out Form
- Write Recommendation Form
  - List of recommended hardware
  - Product List
  - *List of Available New Units*
  - Standard recommendation letter
- List of requested Units
- Check-in New Units form
  - Requested Units Form
  - Outstanding recommendations form
Management information screen
Management Reports
   Report Options Form
Support Website
   Incident details webpage
   Incident Submission webpage
Administer Screen
   Administer Entity Forms
   Confirmation Dialog
   Administer Users Form
   Permissions Form
   PermissionsMatrix

4.6.3 Class diagram
A class diagram of the identified Application Domain Entity Classes is included in the Appendix.

4.6.4 Dynamic models - State Charts
For the non-trivial states of Incidents and (Service-) Units, three state chart diagrams are provided. These can be found in the Appendix.

4.6.5 User interface Navigational paths & mock-ups
Screen mockups have been constructed using JBuilder 9.0, an Integrated Development Environment for generating Java applications. This decision was purely for ease of drawing; the final decision on implementation language has not yet been made.
5 System Design

5.1 Introduction

This chapter documents the System Design Phase of a software project. Whereas the Requirements Analysis documented the ‘what’ part of the requirements, the System Design will focus on ‘how’ these requirements will be met by the system. As mentioned in the report’s introduction BORREL will be developed using an incremental approach. This chapter addresses the functionality of all three increments. First the design goals will be listed (§5.2), which are derived from the non-functional requirements. In the next paragraph (§5.3) describes the results of our research on current software architectures. With these in mind, the global design of the proposed system is discussed in §5.4, while a detailed description of the services offered by the subsystems is given in §5.4.7.1.

5.2 Design Goals

The design goals are based on the non-functional requirements and the application domain unveiled during Requirements Analysis. They have been adapted to fit the System Design phase.

5.2.1 Performance

Hardware
System should run on existing hardware.

Concurrent users
Twenty concurrent users should be supported.

Response time
Response time for data retrieval or submittal commands should not exceed 5 seconds.

5.2.2 Dependability

Robustness
Any invalid user input should be noticed and some kind of feedback will have to be provided to the user prompting him to correct the input.

Reliability
Perceived reliability of the system should be near 100%, or user confidence in the system will drop.

Security
Basic measures should be taken to avoid malicious users from accessing the system through the network or the Internet. Access separation between roles like Administrator, Technician & Helpdesk Employee should be loosely implemented.

Availability
System availability must be at least 95% during office hours.
5.2.3 Cost

**Development cost**
The only development costs IRM will incur are for acquisition of development tools. Constraints on these costs are minimal, and will not be taken into account.

**Training costs**
User training will be in parallel with increment testing, and is therefore spread over an initial period of one week. This minimizes impact on normal operations.

**Maintenance costs**
System maintenance should be possible by ACS staff. Involvement of ACS staff in the system implementation phase will assure familiarization with the system. This will lead to minimized maintenance cost.

5.2.4 Maintenance
Extensibility (ease of adding new functionality) and readability (ease of understanding the system by reading the code) are the most important maintenance criteria. This means that modifiability, adaptability, portability and traceability of requirements will be put in second place.

5.2.5 End user

5.2.5.1 Utility
How well does the system support the work of the user? There are some work-processes that will remain unsupported. Possibilities for future support of these processes will be taken into account during System Design. Unsupported processes are:

- Problem Management
- Configuration Management
- Financials of acquisition process

5.2.5.2 Usability
User Documentation will be written according to university wide standards.

5.2.6 Trade-offs
Some (implicit) design goals conflict with each other. To guide the process of making design decisions, the following trade-offs are proposed.

5.2.6.1 Extensibility vs. simple design
As requirements can change pretty fast and there are a lot of features that can be added to the system, but that are currently outside our scope, the system should be very extensible. Even if this adds complexity to the system.
5.2.6.2 Usability vs. high functionality
Usability is a very important aspect of BORREL, because when we're in the Netherlands again, the system itself is its only ambassador. If some frequent tasks appear to be cumbersome, the system will lose popularity among its users. Usability will be preferred over high functionality.

5.2.6.3 Use of OTS Components vs. a Strict Model-View-Control Design
Of The Shelf (OTS) Components will be used as much as possible, in order to deliver an implementation of the highest quality in the shortest time. Designing a system with OTS Components, however, is often quite a challenge, because the system design has to be integrated with the paradigm the OTS Components adhere to using sophisticated design patterns. Nevertheless, we're convinced that a design that has been well reflected on will be a great advantage during the implementation phase.

5.3 Current software architectures for similar systems
Similar systems were researched for two purposes: to obtain insight in the good practices of HDIS designers and to search for a code base on which we could base our system.
Systems and documentation created during other IPDI projects were very helpful; these are discussed in §5.3.1. We have also investigated numerous other systems, especially those following the Open Source philosophy; these are discussed in §5.3.2.

5.3.1 Software related to similar projects

5.3.1.1 HIST
The Helpdesk ITIL Support Tool (HIST) was developed for the Cochin University in India by two TU Delft students: Mark Dumay and Remco Groeneweg [Dumay & Groeneweg, 2001].
Their implementation focuses on a CMDB and relies heavily on the ITIL guidelines. HIST implements a full graph-oriented CMDB. Incident Management is developed, although there does not seem to be a connection with the CMDB and Problem Management does not exist.
Unfortunately, the bulk of HIST design decisions are not motivated in [Dumay & Groeneweg, 2001]. During a meeting with the two students, however, they have dissuaded us from implementing a CMDB the way they had done, as they regarded it as overkill.
Their system was implemented using MS SQL Server as DBMS and Java as implementation language for both server and client. Java was used because both the students were already familiar with it.
Furthermore it should be mentioned that the requirements that are imposed on their project are just a subset from the BORREL requirements. For example, support for ServiceUnits, acquisition of new Units, KnowledgeBase and FAQ functionality is absent.
Due to the design decisions in 'Design Considerations' it is concluded that HIST is very valuable as a reference, but cannot function as a framework for our system; mostly because of the different requirements, which led to a fundamentally different design.

5.3.1.2 HEAT

Some years before our internship, in 1997, Sybren Eikennar, another TU Delft student, preceded us in his efforts to streamline the End User Computing Support unit, as IRM-HDS was called in those days. He tried to introduce HEAT, an existing HDIS, into the IRM organization. [Eikennar, 1997].

In case you were wondering: the reason we have been at San Carlos is because the HEAT-implementation has been abandoned.

Some drawbacks have already been mentioned in [Eikennar, 1997].

- The configuration management capacities of HEAT are limited. However, this drawback mostly conflicts with the pure ITIL guidelines and not really with our own set of requirements.
- The security options are based on the operations, rather than on the database. This also does not seem a problem to us, as security should foremost be set up to not cause much overhead.
- The version Eikennar used suffered from a lot of bugs. However, it is to be expected that at the moment, six years later, most of these bugs will have been fixed.

As stated, these drawbacks do not directly affect our project. Some more research though, taught us that HEAT is an extensive, bulky and static application, which is not suited for the relatively informal situation, with its specific requirements, at IRM. And of course, there is a huge license fee for the HEAT-system, which a lot of its competitors don't have. In [Dumay & Groeneweg, 2001, blz. 36] is cited: "In Dar-Es-Salaam [...] they first tried to implement the commercial support tool 'Heat'. This implementation completely failed, mainly due to the complexity of the program."

5.3.2 The rest of the playing field

There is a lot of software available in this market; Commercially, but also Open Source.

An Open Source application that suited our needs would be ideal of course, because of its customizability and because no license fees will have to be incurred. Unfortunately, most of these applications focus solely on Incident Management, while Configuration Management or even the notion of a device is not practiced at all. ITIL still appears to be a widely unknown phenomenon in the Open Source community. A few interesting Open Source alternatives were found though, most of them being released under the General Public License (GPL) [GNU, 2003]
5.3.2.1 IRM – The Information Resource Manager
IRM is a web based Helpdesk support-tool that offers both Incident Management and Configuration Management. [Sourceforge, 2000]. The tool is currently in the so-called 'mature' state. The general set-up is according to our requirements, but a lot of additional adjustments have to be made. However, a look at the code shows that it in no way adheres to our structured programming style.

5.3.2.2 Helpdesk Issue Manager
Helpdesk Issue Manager is also a web based Helpdesk support-tool that only offers Incident Management [Sourcefourge, 2003]. CMDB functionality has been vaguely announced, but is not implemented yet. However, Helpdesk Issue Manager is very clear and simple, supports Incident history and a knowledge base. The application has only recently been developed, so its interface looks very fresh and is very nice to handle. Sadly, the code of this program is also one big chaos.

5.3.2.3 Pratis
A helpdesk Issue Manager is also a web based Helpdesk support-tool that only offers Incident Management [Sourcefourge, 2003]. CMDB functionality has been vaguely announced, but is not implemented yet. However, Helpdesk Issue Manager is very clear and simple, supports Incident history and a knowledge base. The application has only recently been developed, so its interface looks very fresh and is very nice to handle. Sadly, the code of this program is also one big chaos.

5.3.2.4 IRCM
IRCM is an Incident Registration and Configuration Management Tool, developed by Eric Giesselbach [Giesselbach, 2003a]. The application meets almost all of our requirements but is not completed yet. Once the application is finished, it will be released under the GPL [GNU, 2003] so we would have full control over the source and it could perfectly function as a foundation for BORREL. Nevertheless, we contacted Eric to explore opportunities for cooperation. Eric showed us a demo-version [Giesselbach, 2003b]. However, due to an agreement with Giesselbach’s employer, which is also involved in the development, the code is classified until the first official release, which is expected to be at the end of august. As this is too late for us, we can conclude that IRCM is also not an option.

5.4 Proposed software architecture
Taking into account the above considerations, we will now sketch the BORREL system design. The system will be a two-tier architecture, with a database residing on the server and two different clients. The IRM Staff Interface client will be implemented in Java (client side logic) and the Customer Interface client in PHP, thus putting business logic and parts of the presentation logic on the server.
In the first paragraph (§5.4.1) design considerations are discussed. Next (§5.4.2), the system is divided into subsystems. In §5.4.3 (Hardware/software mapping) details are given about the client/server-partitioning and the hardware and software platforms that are to be used. Next, 'Persistent data management' (§5.4.4) and 'Access control & security' (§5.4.5) are discussed. Paragraph 5.4.6 describes how global communication within the system is handled and in §5.4.7 some 'boundary issues' such as installation and administration are explored.

5.4.1 Design considerations

A lot of fundamental design-decisions had to be dealt with. The most influential decisions are listed below, including the trade-offs involved and the argumentation regarding our choice.

5.4.1.1 Incident Management or Problem Management?

Our compromise: Only implement support for Incident Management without support for Problem Management

Rationale: ITIL clearly distinct two different processes with respect to the handling of incidents:

- Incident Management, which concerns the handling of the incoming incident: register the incident, try to identify the root-cause and to resolve the incident over the phone, inform the customer of the status of his reported incident.
- Once the Incident appears to be caused by something that has to be attended by the technical department (a Problem), the Problem enters the Problem Management track. This way, multiple Incidents can be caused by one Problem.

We have decided not to implement Problem Management, because many interviews taught us that the bulk of the Incidents is isolated. Implementing a separate Problem Management track would add a lot of extra complexity to our system, while not adding much extra value (See trade-offs).

5.4.1.2 CMDB as specified by ITIL or a customized solution?

Our compromise: Loosen the strict recommendations ITIL imposes upon a CMDB. The procedure ITIL suggests is as follows: First all hardware, software and installation requests, the so called Configuration Items (CI’s), should be added the CMDB, relations between CI’s should be added. All CI’s should be physically labeled. After that, when the system is operational, the CMDB has to be maintained, meaning CI’s en the CI relations can be added, deleted and modified. ITIL emphasizes the completeness and correctness of the database.
Our solution includes a variation on configuration items by introducing the concept of 'Units'. The system won't know of any Units when it starts operating. During operation service-units, new units and pulled-out units will pass the Help Desk. On this occasion the unit will be added, modified or deleted in the database. No Unit relations will be stored. Besides Units we will also define UnitTypes. When a Customer reports an Incident that's related to a Unit without an id, only its type will be recorded.

**Rationale:** With respect to the benefits of the ITIL guidelines, these are the reasons why we came up with our own customized solution:

- We learned from [Dumay & Groeneweg, 2001] that implementing CI relations is very complex and time consuming, while the benefits are negligible. We will therefore not support CI relations in our system. Setting up and maintaining a CMBD (by ITIL guidelines) would be very time consuming for the help desk staff and it would be very hard to keep the CMDB complete and correct.
- Working procedures should have to be changed drastically, which is very hard to realize in eight weeks or less in this culture.
- With our approach, the Unit of the customer refers to may or may not be in the database, something that the ITIL-guidelines do not allow, because this conflicts with the requirement of completeness. For us, this is absolutely no problem, because we only record Units in relation to Incident Management.
- IRM will steadily harvest information about USC IT-infrastructure, making the system more valuable every day, without obliging the IRM employees to spend weeks on filling a huge database.
- Should support for a fully-fledged CMDB be deemed necessary in the future, the system structure may be easily extended to include it. In this case, existing data will need to be checked for validity.

### 5.4.1.3 Installations of hardware and software

**Our compromise:** As mentioned in the requirements of the system, installations of hardware and software must also be handled by system. Like ITIL we will regard an installation as an Incident.

**Rationale:** In practice both are handled similar.

### 5.4.2 Subsystem decomposition

Initial subsystem decomposition is derived from functional requirements. In an iterative process, the decomposition is refined to take into account hard- & software mapping, persistent data management, access-control, security, software control and boundary conditions. The resulting set of subsystems is described below. Contained objects and all subsystem relationships are depicted in the appendix.

Acquisition, Configuration, Incident and Report subsystems communicate directly with the server-side DBMS to obtain their data. They employ sophisticated caching mechanisms to ensure speedy and reliable data access.
All subsystems reside on the client, except for the WebGUI subsystem. The latter will be on the web server, working in close harmony with the Customers web browser.

| **Acquisition** | Supports the acquisition process of new parts. On arrival of the parts they are handed over to the Configuration subsystem |
| **Configuration** | Models the equipment and software that exists within the organization. Among other things, maintains information on Unit status and whereabouts, as to accommodate the repairing process. |
| **GUI** | Delivers the functionality of all other subsystem to the IRM Staff in an understandable, task oriented way. It is based on the JBuilder JDataExpress (Swing) components. |
| **Incident** | Handles incident resolving and -logging processes. It depends heavily on Personnel, Configuration and Acquisition subsystems. |
| **Report** | Can generate various reports for viewing and/or printing. These include gate passes & management information reports. |
| **WebGUI** | Controls the presentation and functionality of the system for the Customers. |
| **Personnel** | Deals with user and access management. Exposes authentication and authorization functionality of the DBMS, and maintains the skill-set of Technicians. |

5.4.3 Hardware/software mapping

This paragraph discusses how the software architecture can be achieved by mapping it to hardware and software components. First, it is inventoried what hardware and software is currently already present. Then, the current experience of the ACS staff is described. Third, the network connectivity in the relevant environment is summarized.

Since all relevant knowledge has been explicated, the system design is now partitioned in a client- and a server side and implementation platforms are chosen.

5.4.3.1 Available server hardware / software

See Appendix.

5.4.3.2 Available client hardware / software

Most client computers university wide are x86 family PC's that run on MS Windows 98 and up. Internet Explorer 4.0 and up is installed on the workstations. At the IRM-ACS every staff member has a Pentium III 500 MHz, 128 MB RAM. At IRM-TNS & HDS not every staff member has his or her own computer. There are 7 computers for 11 technicians. In future when IRM has offices at all campuses every technician will have its own computer. The Helpdesk Employee uses a Pentium III 451 MHz, 128 MB RAM.
5.4.3.3 Current information systems

<table>
<thead>
<tr>
<th>Software</th>
<th>Programming language</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARIS (Academic Register Information System)</td>
<td>Delphi</td>
</tr>
<tr>
<td>FIS (Financial Information System)</td>
<td>PowerBuilder</td>
</tr>
<tr>
<td>HRMIS (Human Resource Management Information System)</td>
<td>PowerBuilder</td>
</tr>
<tr>
<td>LIBIS (Library Information System)</td>
<td>(Externally developed)</td>
</tr>
</tbody>
</table>

Table 2: Information Systems currently in use

5.4.3.4 Current ACS platform experience

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Michelle Casana</td>
<td>Head</td>
</tr>
<tr>
<td>Joeffrey Aquino</td>
<td>ARIS and HDIS programmer in-charge</td>
</tr>
<tr>
<td>Karl Bajenting</td>
<td>FIS programmer in-charge</td>
</tr>
<tr>
<td>Bianca Rose Vallejos</td>
<td>HRMIS programmer in-charge</td>
</tr>
<tr>
<td>Garry Patigayon</td>
<td>System administrator</td>
</tr>
</tbody>
</table>

Table 3: Functions of the IRM-ACS staff

<table>
<thead>
<tr>
<th>Name</th>
<th>Java</th>
<th>Delp hi</th>
<th>C++</th>
<th>VB</th>
<th>PowerBu ilder</th>
<th>MS SQL</th>
<th>MyS QL</th>
<th>ASP</th>
<th>PHP</th>
<th>HT ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Michelle Cansana</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Joeffrey Aquino</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Karl Bajenting</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Bianca Rose Vallejos</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Garry Patigayon</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Competences of IRM-ACS staff

IRM-ACS has licenses for the following development software:
- PowerBuilder
- Delphi

5.4.3.5 Network connectivity

The figure below illustrates the network connectivity specifically for the IRM department, but any other department with a network is modeled the same way. As you can see some workstations have an Internet connection, while others have not. It is not possible to have both an Internet- and an intranet connection.
5.4.3.6 Client/Server architecture

*Our compromise:* two-tier architecture with a fat client for all employee interfaces (HE, Technician, Head, Administrator) and a thin client for the customer interface. All concurrency issues will be handled by the DBMS.

*Rationale:* We choose a two-tier architecture because of the relatively limited computing and storage demands of the system. A full-blown three-tier setup is regarded as overkill. So let's have a look at the possible two-tier client/server-designs next.
Figure 2: Different Client/Server architectures

Figure 2 depicts the common ways to partition an application into a server and a client. The clients on the left will be referred to as 'fat' clients, because of their complexity and the clients on the right will be referred to as 'thin' clients.

We will now discuss the partitioning we chose for the client the employees use and the client the Customers use.

**IRM Staff Interface**

*Our compromise:* For the IRM Staff Interface we will employ the second architecture, where all business logic is integrated in the client and the server only holds the DBMS. *Rationale:*

- By integrating all presentation logic in the client, rich interfaces become possible, including continuously up-to-date views, live input-validation, etc. This is not possible in a ultra-thin-client (See below) ➔ Rich GUI
- The only things the server has to handle in this set-up are access control and concurrency. As these can both be handled by the DBMS, no further server logic is necessary. This makes it possible to delegate all network communication to a database communication protocol such as ODBC. ➔ Simplicity
- Computing power of the workstations is sufficient to handle a fat client. (See §5.4.3.2: Available client hardware / software). ➔ Compliance to speed requirements
- Client-software needs only to be installed locally on a few computers, so clients can be easily installed and maintained. ➔ Compliance to maintainability requirements
- As the application will only be reachable through the local network, the fact that access control now has to be implemented at database-level (See §5.4.5: Access control and security) is not a very big issue. ➔ Compliance to security requirements
Customer Interface

Our compromise: The Customer Interface will employ the rightmost architecture (ultra-thin client), where the server contains the DBMS and the business logic. It also performs most presentation logic tasks, by generating HTML-code. The client only needs to parse the HTML, show the GUI to the user and accept user input.

Rationale:

- The application can be reached from any computer with a browser, through most firewalls (via http), without the need to install any client software. This is a huge advantage, because it should be possible to access the Customer-client from every computer at USC. → Accessibility
- Computing demands for the client workstation are low, which is also a big advantage, because hardware specs for Customer hardware are very heterogenic. → Speed
- Security-logic can be implemented at the best layer: the business logic layer. (See §5.4.5: Access control and security) This is an advantage because the server the Customer-client uses is accessible from the whole USC network, so security is a biggest issue then for the employee-client.

5.4.3.7 Implementation platforms

BORREL will use:

Java for implementing the IRM Staff Interface
- Because of its sophisticated Object Oriented (OO) support
- Because of the integration with Rational Rose, which allows generating code from models and updating models from code.
- Because a lot of standard components and nice tutorials are available
- Because all participants are familiar with Java

Microsoft SQL Server as DBMS
- Because it's already available at IRM, so no license or installation efforts are needed and technical staff already knows how to maintain it.
- Because of features like support for relational integrity

PHP for implementing the Customer Interface
- Because it easily allows the creation of a web-based application
- Because it is much simpler to setup than a system based on servlets
- Because it can communicate with the same SQL Server the IRM Staff Interface talks to
- Because all participants are familiar with PHP

JDBC for MS SQL Server to connect to the database
- Because it connects to the database in a convenient way, without the need for a JDBC-ODBC-bridge

DataExpress Component Library as Data Access Framework
- Because connection pooling and caching is already implemented.
- Because the link between a User Interface component and the database is very easy to implement
- Because the components are integrated in the JBuilder IDE
The DataExpress Component Library (DECL) is offered by Borland JBuilder. We use two components of it. The DataExpress component contains packages that provide classes for data connectivity, management and manipulation through JDBC and utilities for input validation. DbSwing provides Swing components capable of accessing database data through DataExpress.

5.4.4 Persistent data management
All data will be stored in a database because of the large quantity and the complex data retrieval operations that should be supported. An object-oriented database might be considered, but as at this moment no OO databases that are both mature and affordable exist, a relational database will be used.

5.4.5 Access control and security

5.4.5.1 Access control
The access control mechanism should be as simple as possible, yet flexible enough to deal with future organizational changes. Four distinct roles will be defined, while users can fulfill one or more roles. These roles are, with the most powerful roles listed first:

- Administrator: Allowed to do everything
- Head: Allowed to perform some unique actions that require a certain level of authority
- HE: Allowed to do almost anything except some actions that are exclusively allowed by the Head and some administrative actions.
- Technician: Allowed to check in/out Units and register his/her own actions.

More roles can be added, so extension and fine-tuning of the access control is possible, without the need to modify the code.

Users gain access to the system by entering their unique username and password combination.

5.4.5.2 Security
Security should never be fully handled on the client side, because network-traffic is unsafe. Sniffing can be avoided by using a secure channel, but there are still ways in which a user can construct its own client application that does not do any authentication or authorization at all. Security in our system thus resides on the server.

IRM Staff Interface
While the client application mediates in the authentication process, actual authentication will be handled server-side by the DBMS. After authentication, the DBMS restricts access to certain data based on authorization rules. The GUI will also be adjusted according to the authorization rules by showing or hiding individual components. The latter is purely a cosmetic feature and cannot be abused to compromise the security of the system.
Customer Interface
As both interfaces connect to the same database, authorization is already performed by the DBMS. Adjustment of the GUI to the current user with its respective permissions is done the same way as for the IRM Staff Interface, except now it happens on the server.

Because the Customer Interface has to be much more widely accessible, a problem could occur if someone tries to sniff an unencrypted username/password combination. The remedy is to use HTTPS (Secure HTTP), in combination with a security certificate.
We will not implement this, however, and leave it as a recommendation.

5.4.6 Global software control
Three important mechanisms will be discussed. The overall integration of the DataExpress Component Library in the design; the way input validation takes place; and the functionality behind the Fully Object Oriented Database Synchronized Abstraction Layer (FOODS Abstraction Layer or just FOODS)

5.4.6.1 Integration of DataExpress Component Library

Figure 3: Software Control regarding the integration of the DataExpress Component Library

We will consider the internals of the data provision by looking at unitTable, an instance of a JdbComponent. The journey starts at MainControl, which owns the reference to Database and passes it to any subcontroller it creates, so also to ConfigurationControl, the relevant subcontroller in our case.
When a JdbComponent is created, it needs data to display.
For this purpose, ConfigurationControl returns a QueryDataSet, which is then linked to unitTable. The QueryDataSet is actually constructed by ConfigurationStorageControl, which communicates with Database by means of SQL-statements.
The QueryDataSet functions as a GUI-model and provides the JdbComponent with an up-to-date view on the data it needs. This view can even be editable, while the QueryDataSet takes care of the necessary SELECT, UPDATE, INSERT and DELETE queries to ensure continuous synchronization between the QueryDataSet and the database-tables involved.

JdbComponents know how to treat a QueryDataSet and can present the data to the user through any visual representation one likes.

### 5.4.6.2 Validation of user input

![Software Control regarding the validation of user input](image)

A mechanism has to exist to check the user input. When the user is asked to enter an e-mail address, only one '@' should be entered and at least one dot. When a date is entered, it should be within some context-sensitive range. These and a lot of other things have to be checked. As you can't have the database do any more than just basic checks, the validation has to occur within the business-logic.

As explained before, each GUI component gets its data from a QueryDataSet. So this is the place where we will check our input.

As soon as an attribute of a QueryDataSet is being updated, it notifies all classes that have been registered as ColumnChangeListener.

This is where the Validator-class comes in. Every database-table has its own Validator extension, which contains all the table-specific validation-logic. One or more Validator subclasses listen to QueryDataSet and check whether all the input rules have been adhered. If not, they throw a ValidationException, which causes the whole updating-process to be rolled back.

### 5.4.6.3 The Fully Object Oriented Database Synchronized Abstraction Layer
Figure 5: Software Control regarding the FOODS Abstraction Layer

Because the JdbComponent's can handle QueryDataSet's, which are like recordsets, very well, we just feed them QueryDataSet's. As we have relatively little to do with the whole communication with the JdbComponent's, we don't care that QueryDataSet's are not pure OO, yet very effective. However, for the parts of the system that need more custom logic to operate than just the mechanism described in §5.4.6.1, we've decided to design a 100% OO interface to the database. The idea is that the attributes of a class just refer (by the attribute's get- and set-method) to a QueryDataSet. Remember that a QueryDataSet in its place refers to the database by converting method-calls into queries.

To be able to refer to the right columns we need a QueryDataSet that refers to a 'standard' select query to the table that corresponds to the class (SELECT * FROM classname).

A problem that occurs is that QueryDataSet returns specific rows by value rather than by reference. A Synchronizer that listens to PropertyChangedEvent's of both the QueryDataSet and the object in question causes them to be synchronized.

5.4.7 Boundary conditions

5.4.7.1 Initialization

The system is started by starting the MSSQL Database Engine. To use the system, the Java-client is required. This client is distributed using the Java Webstart framework, which is also suited for roll-outs of future updates.

MainControl will be the first object to be initialized. It will make a connection with the database. It then launches GUI components as needed, and provides these components with the various subsystem control objects that they depend on.
5.4.7.2 System Administration

Some of the entities in the system require administration. Administration actions needed, prepended with their priority, are:

- (1) add, modify and delete users and their rights
- (1) Appoint service units
- (3) add, modify and delete configuration items
- (1) add, modify and delete configuration item types
- (1) add, modify and delete departments/campuses/etc
- (1) add, modify and delete SLA specifications
- (3) add, modify and delete skills
- (3) add, modify and delete user permissions

The following generalized administration use cases describe the flow of control for aforementioned functionality.

<table>
<thead>
<tr>
<th>Use case name</th>
<th>Participating actors</th>
<th>Entry condition</th>
<th>Flow of events</th>
</tr>
</thead>
</table>
| Administer          | Administrator (initiator) | Administrator wants to add, modify or delete an employee, a unit, a configuration type (item), a location, a SLA specification, a skill or a user permission. | 1. Administrator performs the LogOn use case  
2. Depending on what the Administrator wants to do he activates one of the following use cases  
   a. AdministerGeneral  
   b. AdministerRightsMatrix  
3. Administrator performs the LogOff use case  
   Administrator knows what to do next |

<table>
<thead>
<tr>
<th>Special requirements</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AdministerGeneral</td>
<td>Administrator (initiator)</td>
<td>Administrator wants to add, modify or delete one of the following entities: a user, a unit, a configuration type (item), a location, a SLA specification or a skill.</td>
</tr>
</tbody>
</table>
Flow of events

1. Administrator opens his *Administer Screen*.
2. He selects the entity he wants to administer.
3. An *Administer Entity form* shows up. It shows the instances and an *Add*, *Delete* and *Modify* button
   a. When adding or modifying an instance, an *Entity Form* shows up. Administrator can insert or update an instance.
   b. When deleting an instance a *Confirmation Dialog* show up and Administrator can confirm or cancel the deletion process.
4. Administrator closes the *Administer Entity form*.

Exit condition

Administrator knows what to do next.

Special requirements

5.5 Subsystem services

Subsystem services are derived from the functional requirements and the use cases. Each service is responsible for (a piece of) a requirement or use case. When they are combined they define the full system functionality.

5.5.1 Acquisition

Check In New Units
(Un)postpone Recommendations
Modify Acquisition Recommendations

5.5.2 Configuration
Check In Units
Check Out Units
Check Out Service Units
Register Fixed Pull Outs
Register Irreparable Pull Outs
Appoint Service Units
List all units

5.5.3 GUI
Deliver all other subsystem functionality to the user in an understandable way.

5.5.4 Incident
Assign Incident
New Incident
Get Assignments
New Installation Request
Get SLA
Get Customers
Get Incidents
Modify Customers

5.5.5 Personnel
Get Technicians
Log off
Log on
Issue Right
Revoke Right
Modify Campus
Modify Department

5.5.6 Report
Create and print various reports
Implementation

Of course the result of the implementation phase is the software program BORREL itself. The reason why there is a chapter implementation in this report is that while implementing we encountered a few problems we hadn’t foreseen during the analysis and design phases. Since the RAD and SDD were already approved we decided not to update those documents but mention them in a separate chapter. So here they are.

6.1 Synchronization and notification

How to notify the client when changes in the database are made by another client (or the server)? For the moment IRM only uses one client program which makes changes in the database so this not a big problem. Later when multiple clients are running there has to be a solution for this.
7 Conclusions

In this chapter you find a brief overview on the different parts of the project and what we can conclude from them.

7.1 RAD and SDD

The main objective is to develop and implement a tool for supporting the Helpdesk department. So the first thing we did was to get a picture of the current procedures. Most of the information we gained during interviews with the head of the department, several helpdesk employees and technicians. In this way we met a lot of people and it was our first contact with the local culture. We had some difficulties during the first interviews on how to get the right information out of the people. We also noticed it is hard to get an opinion or suggestion out of the locals, while that was what we really needed. But after a week we finished the interviewing having a good view of the processes. The problems we encountered we blame to the cultural gap (hierarchical versus flat). Based on the information of the interviews and documentation we made a proposal what will later become BORREL. After approval of the RAD we started working on how to realize the system. We investigated how to integrate the system in the department. We chose the software architecture, split the system in subsystem and made a global design.

7.2 Implementation

The SDD was the base for the implementation. We only finished the first increment although we scheduled three. For this we can blame the OTS components we used, documentation was poor and we encountered lots of bugs. But since we used the incremental approach we have delivered a fully working program.

7.3 Passing on BORREL to IRM-ACS

At the beginning we were informed that two programmers will be responsible for BORREL after we left. Halfway the project it was announced the two guys were quitting their job in a few weeks, so passing on BORREL to them would be useless. The other three staff members of IRM-ACS were overloaded with work so passing on the system to them was very hasty and not good. These are the reasons why we put a lot of effort in documenting the system and in this way providing future staff members the required knowledge to maintain and expand BORREL.
8 Recommendations

8.1 General recommendations

Throughout the project many observations have been from which the department or system could benefit. Where possible, these ideas have been incorporated into the project, but some remain unattended. These have been gathered beneath as recommendations.

1. A comprehensive FAQ can be developed by analyzing incident history. Encouraging self-help using this FAQ can help lower the number of incidents that needs to be handled by IRM, and thus lower the strain on the support staff.
2. Increasing technician productiveness by having each one develop specialties through experience can speed up processing time for a problem.

8.2 Specific development recommendations

The following are specific recommendations for the development team.

1. When the web-interface gets more powerful functionality, it is wise to take extra security measures into consideration, as discussed during System Design.
2. The IncidentState is currently mostly set manually. As new functionality causes more information to be gathered for each Incident, the state should be derived automatically where possible, as to avoid inconsistencies.
3. A feature that’s fairly simple to implement at this stage, but nonetheless very useful is to have the system keep track of who first entered an incident.
## References

<table>
<thead>
<tr>
<th>Reference</th>
<th>Author(s)</th>
<th>Title</th>
</tr>
</thead>
</table>
Appendix A - Requirements analysis

1 Current System

1.1 Flow Charts

The different types of calls will be followed up by different processes. These are described in the flowcharts below.

Figure 6: Legend of the flowcharts
Customer C stimulates the helpdesk

Helpdesk-employee H catches stimulus

H logs the stimulus in the Help Desk Info Log Sheet and marks it red

Is the stimulus an ACS software Incident, an Installation Request, or a Technical Incident?

H tries to resolve the Incident over the phone

See flowchart 'Installation Request'

Technical Incident

The problem is handed over to a member of either the technical or network staff T (preferably Winston) so T can try to solve the problem over the phone

Succes?

Yes

No: problem

The problem is handed over to a member of either the technical or network staff T (preferably Winston) so T can try to solve the problem over the phone

Succes?

Yes

No: someone has to go to the site

Is the site remote?

Yes

No: wait until

Is someone (T2) available at the office?

Yes

Call or text T2

H marks problem black in logfile & status (remark) is updated

Wordt deze reassigning ook ok langs geadmireerd?

See flowchart 'ACS Software Incident'

Site remote?

Yes

No

Is someone (T2) already present at the site? Check the 'Help Desk Info Log Sheet'

Yes

No: wait until

- A standard gatepass is printed, the required info (T2's name, destination), is entered and the gatepass is signed by Rhea.
- A Call Ticket is also printed

The 'Help Desk Info Log Sheet' is updated. (attended by)

- T2 heads for the troublezone

T2 tries to solve the problem on site

Flowchart 1: Incoming Technical Incident Stimulus 1/2
Flowchart 2: Incoming Technical Incident Stimulus 2/2
Flowchart 3: Incoming ACSS Incident Stimulus

 Boris tells an employee E from IRM-ACS

 E fixes the problem (if necessary contact customer C)

 E tells Boris the problem is fixed

 Boris marks the problem in the 'Helpdesk log sheet' as fixed
Flowchart 4: Incoming Installation Request Stimulus
2 Proposed System

2.1 Use Case Model

![Use Case Model Diagram]

- **RegisterUnitsArrived**
- **PostponeRecommendation**
- **Head**
- **ViewReport**
- **QueryStatusThroughWWW**
- **NewIncidentThroughWWW**
- **Manager**
- **Customer**
State Chart 1: Unit
State Chart 2: Incident

Mark Unit as Service Unit

Stocked

CheckOutServiceUnit

In Use

Demark Unit as Service Unit

State Chart 3: Service Unit
Appendix B – System Design

1 Two Tier vs. Three Tier

There are two widely used software architectures for systems like ours. These are the ‘Two Tier’ (client/server) architecture and the ‘Three Tier’ architecture. The descriptions are quoted from [Sadoski, 2000].

1.1 Two tier architectures

With two tier client/server architectures (see Two Tier Software Architectures), the user system interface is usually located in the user's desktop environment and the database management services are usually in a server that is a more powerful machine that services many clients. Processing management is split between the user system interface environment and the database management server environment. The database management server provides stored procedures and triggers. There are a number of software vendors that provide tools to simplify development of applications for the two tier client/server architecture [Schussel 96, Edelstein 94]. The two tier client/server architecture is a good solution for distributed computing when work groups are defined as a dozen to 100 people interacting on a LAN simultaneously. It does have a number of limitations. When the number of users exceeds 100, performance begins to deteriorate. This limitation is a result of the server maintaining a connection via "keep-alive" messages with each client, even when no work is being done. A second limitation of the two tier architecture is that implementation of processing management services using vendor proprietary database procedures restricts flexibility and choice of DBMS for applications. Finally, current implementations of the two tier architecture provide limited flexibility in moving (repartitioning) program functionality from one server to another without manually regenerating procedural code. [Schussel 96, Edelstein 94].
1.2 Three tier architectures

The three tier architecture (see Three Tier Software Architectures) (also referred to as the multi-tier architecture) emerged to overcome the limitations of the two tier architecture. In the three tier architecture, a middle tier was added between the user system interface client environment and the database management server environment. There are a variety of ways of implementing this middle tier, such as transaction processing monitors, message servers, or application servers. The middle tier can perform queuing, application execution, and database staging. For example, if the middle tier provides queuing, the client can deliver its request to the middle layer and disengage because the middle tier will access the data and return the answer to the client. In addition the middle layer adds scheduling and prioritization for work in progress. The three tier client/server architecture has been shown to improve performance for groups with a large number of users (in the thousands) and improves flexibility when compared to the two tier approach. Flexibility in partitioning can be as simple as "dragging and dropping" application code modules onto different computers in some three tier architectures. A limitation with three tier architectures is that the development environment is reportedly more difficult to use than the visually-oriented development of two tier applications [Schussel 96, Edelstein 94]. Recently, mainframes have found a new use as servers in three tier architectures (see Mainframe Server Software Architectures).