

Use Case Experiment

Solution proposed by:

Working Group Software Engineering
University of Heidelberg, Germany

Prof. Dr. Barbara Paech
Alexander Delater
Robert Heinrich
Rumyana Proynova



1. Introduction

In this case study we have applied our method TORE (Task-oriented Requirements Engineering)¹. The essential difference to usual use case approaches is to start from a system independent user task description. These user tasks are refined into use cases by looking in detail at domain data and workspaces which capture a high-level view on the user interfaces.

2. Overview

In the following we provide

1. A list of assumptions we made which were not explicitly stated in the specification. These are assumptions were triggered by questions which had to be answered to come up with the TORE descriptions.
2. A usage diagram giving an overview on the user tasks and use cases and the actors involved. The user tasks are not detailed in the following for reasons of space.
3. A domain data diagram capturing the most important terms used in the use cases.
4. The use cases. We use the following notation for the description of the UC steps. Each actor step is labeled Ai and each system step is labeled Sj. The order of execution is from left to right and top-down. Optional steps can be carried out in any order. The steps marked with [optional *] can be repeated many times. If a step marked with [optional X] is carried out once, no further optional step of this use case can be carried out. VARj describes different ways of performing a step.
5. The workspaces. Workspaces structure the data and functions into subset which are provided to the user coherently. In general a workspace will correspond to several views on the user interface because not all information of one workspace can be captured on one screen. For this case study only few views for each workspace are sufficient. Therefore we describe the workspaces by screen shots. However, in a real project there are reasons to not use directly the screen shots because at this stage screen details do not yet reflect usability criteria. So the specific layout is misleading.

3. Assumptions

- User names are not unique. Therefore, the user must submit his or her e-mail address (we assume that this is unique not only for a given moment, but also that old employees' addresses aren't recycled). The system reads the employee's name and phone number from the company's employee database.
- A user may want to reopen an old request, if a problem that appeared solved reemerges.
- A request is complete, if it contains the data listed in the business rules.
- When a new request is entered, it automatically has the status "first line". If it cannot be handled immediately, the status has to be changed by a first line supporter to "second line".
- The user can add information to the problem description field. S/he isn't allowed to delete existing information in that field or to change other fields.
- The user is allowed to close his/her request.
- A supporter can forward a request to a colleague (an expert) or release it for a whole line.
- A supporter is allowed to close a request without solving it, when solution is impossible or not wanted by the organization.
- In the "requests" workspace, the supporter can see a list of all requests ordered by any parameter
- The management wants to be able to filter statistics for given parameters (e.g. the management wants to see a list of all requests filed in a given time period)
- The management wants to utilize the statistics independently of the hotline system, so it needs an export function for different formats (e.g. print on paper, export to an Excel sheet, etc.)

¹ B. Paech and K. Kohler. Task-Driven Requirements in Object-Oriented Development. Perspectives on Requirements Engineering 753 (2003), 45-67.

- To address the problem with the supporters not entering minor requests, the system will feature a single-click possibility for the entering of minor requests. It will automatically record the time and the supporter, and assign the category "minor" and the status "closed". We assume that it is acceptable that all other data of the request don't get recorded, as the alternative is no record at all.
- The expected closing date isn't used in determining which requests are orphaned. However, the supporter can record it for the convenience of the IT user.
- We assume that when an actor starts a use case in a given workspace, he/she has authorization to use the workspace, because he/she is already logged in. We only mention a log in or a log out where it is essential for the use case flow.

4. Usage Diagram

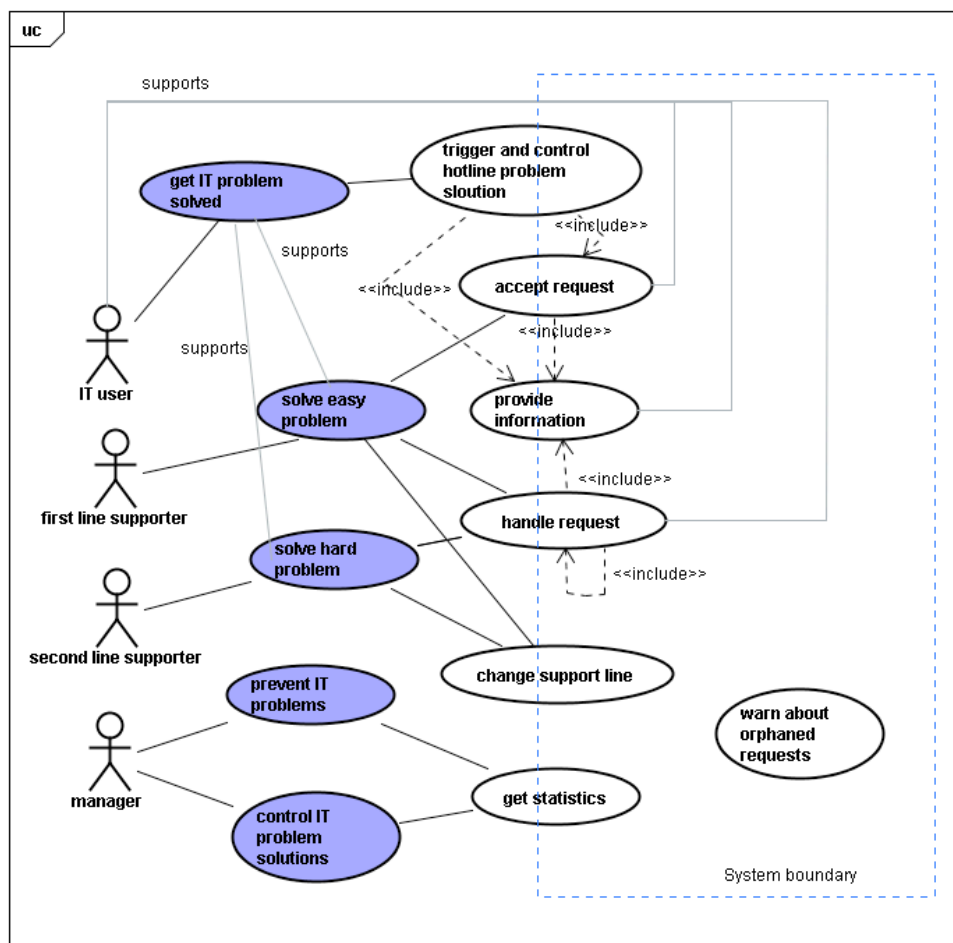


Figure 1: Usage Diagram (blue bubbles are user tasks, white bubbles are use cases)
 Note: The use case which does not involve an actor is carried out by the system autonomously.

5. Data model

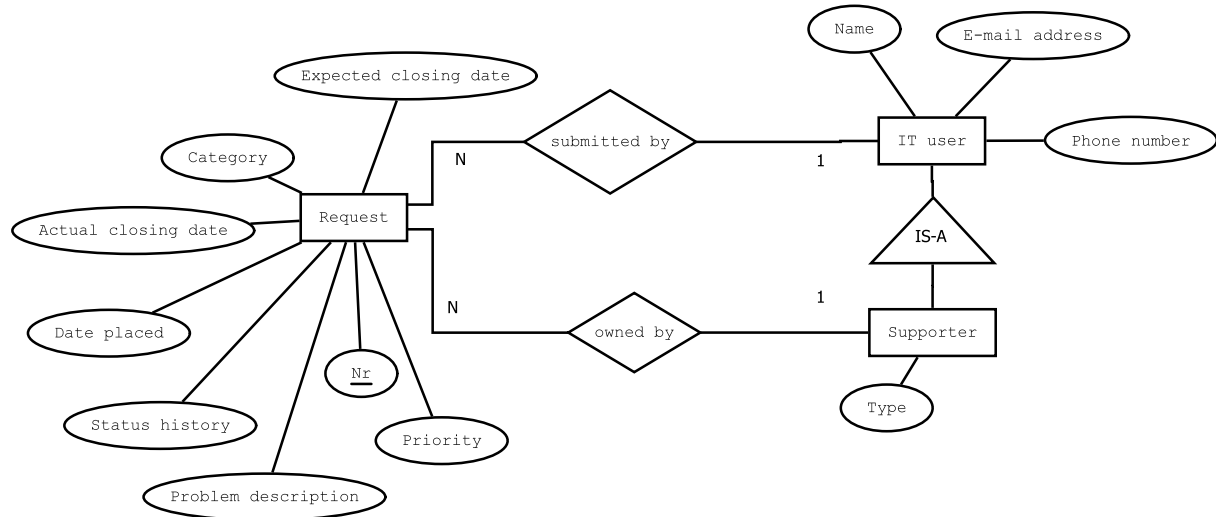


Figure 2: Data Model (usual entity-relationships diagram notation)

A list of the possible statuses:

- **First line:** A first-line supporter must take on the request, for instance because it just arrived.
- **Second line:** A second-line supporter must take on the request.
- **Taken:** The request is handled by a supporter (the owner). The owner may change from one line to another while he is handling the request.
- **Parked:** The request awaits something, for instance an external delivery, and hotline need not do anything meanwhile.
- **Closed:** The request has been handled. However, it may be opened again, for instance because the user doesn't think the problem has been solved.
- Open requests are those that are neither parked, nor closed.

6. Use Cases

Name	Trigger and control hotline problem solution	
Actor	IT user	
Supporting Actors	None	
Goal	The hotline knows about the request.	
Precondition	The user encounters a problem. [Workspace: none]	
Description	Actor	System
	A1) The actor places a new request or reopens an already closed request electronically [Exception: not possible to place/reopen request electronically].	S1) VAR1: The system receives a new request, checks the information for completeness and adds the user data from the employee data base [Exception: information not complete]. It sets the status of the new request to "first line" and the category to "Unknown" and sends the actor an acknowledgement [System function: open] VAR2: the system sets the status of the reopened request to "first line" and sends the actor an acknowledgement [System function: reopen].
	A2) [optional *] [include UC Clarify request]	
	A3) [optional *] The actor views the request to check the status.	S3) The system provides the information on the request [System function: show request details].
	A4) [optional X] If the actor solves the problem himself/herself, he/she closes the request.	S4) The system sets the status of the request to "closed" and records that it was closed by the user [System function: close].
Exceptions	<ul style="list-style-type: none"> - [Not possible to place/reopen request electronically]: If the actor is not able to place or reopen the request electronically, the actor places or reopens the request by telephone or personally [include UC Accept Request]. - [Information not complete]: If the request is incomplete [R1] the system sends a warning. The actor completes the missing information [Proceed to S1] or cancels. 	
Rules	R1: The request is complete when user name, user email, subject, problem description and date placed are given. R2: A new request or a reopened request always gets the initial state "first line".	
Quality Requirements	None	
Data, Functions	<ul style="list-style-type: none"> - Data <ul style="list-style-type: none"> - User data - Request data - System Functions <ul style="list-style-type: none"> - open - reopen - show request details - close 	
Postconditions	The system recorded the request and all necessary information.	
Included Use Cases	<ul style="list-style-type: none"> - Clarify request - Accept request 	

Table 1: Use Case „Trigger and control hotline problem solution“

Name	Accept request	
Actor	First line supporter	
Supporting Actors	IT User	
Goal	The request is accepted	
Precondition	[Workspace: request]	
Description	Actor	System
	<p>A1) actor receives IT help request from IT user</p> <p>VAR1: IT user reports new problem by phone VAR2: IT user reports new problem in person VAR3: IT user asks for reopening of old request by phone VAR4: IT user asks for reopening of old request in person</p>	
	<p>A2) actor enters request details</p> <p>VAR1) if user reported a new problem, the actor enters all data VAR2) if user asked for reopening of old request, the actor reopens the old request and updates its data VAR3) if the user reported a minor problem which can be solved very quickly, the actor only records a minor request, without entering the request data into the fields.</p>	<p>S2.1) system checks data for completeness [Exception: information not complete] S2.2) If VAR1: system stores data and sends an acknowledgement [System function: open] If VAR2: system reopens old request and sends an acknowledgement [System function: reopen] If VAR3: system records a new request of the category "minor". It automatically sets the category to "minor", the date placed and the expected closing date to the current date, and the owner to the actor. The remaining fields remain empty. [System function: open] Then it automatically closes the request. [System function: close]</p>
	<p>A3) [optional X] If the actor cannot handle the request immediately, s/he changes its status to "second line"</p>	<p>S3) The system sets the request status to "second line" [System function: update request data]</p>
Exceptions	[Information not complete]: If the request is incomplete [R1] the system sends a warning. The actor gets the missing information from the user and completes it [proceed to S1] or cancels.	
Rules	<p>R1: The request is complete when name, user email, problem description and date placed are contained. R2: A new request always has the status „first line“</p>	
Data, Functions	<ul style="list-style-type: none"> - Data <ul style="list-style-type: none"> - User data - Request data - System functions <ul style="list-style-type: none"> - open - reopen - close - update request data 	
Postconditions	The system recorded the request and all necessary information.	
Included Use Cases	None	

Table 2: Use Case „Accept Request“

Name	Clarify request	
Actor	First line support, second line supporter	
Supporting Actors	IT user	
Goal	The request is clarified	
Precondition	[Workspace: request] The supporter needs additional information to solve the problem.	
Description	Actor	System
	A1) [optional *] The actor asks the IT user for additional information. VAR1) the actor asks the IT user for additional information by phone VAR2) the actor asks the IT user for additional information by e-mail VAR3) the actor asks the IT user for additional information in person	
	A2) [optional *] The actor asks a previous owner of the request for additional information VAR1) the actor asks the previous owner for additional information by phone VAR2) the actor asks the previous owner for additional information by e-mail VAR3) the actor asks the previous owner for additional information in person	
	A3) the actor or the IT user enters additional information VAR 1) The IT user inserts additional information to the request by himself/herself electronically. [Exception: not possible to provide information electronically] VAR 2) The actor inserts additional information to the request by himself/herself electronically.	S3) The system records the additional information. [System function: update request data].
Exceptions	[Not possible to provide information electronically]: If the IT user is not able to provide the information electronically, the IT user provides the information by telephone or personally [Proceed To VAR2].	
Rules	R1: The user is only allowed to provide additional information to the description. He/she is not allowed to change other fields. He/she is not allowed to delete information in the description field. The IT supporter is allowed to make any changes to following fields: status, priority, category, expected closing date, problem description.	
Quality Requirements	None	
Data, Functions	System functions - update request data	
Postconditions	The system recorded additional information	
Included Use Cases	None	

Table 3: Use Case „Clarify request“

Name	Handle request	
Actor	Supporter (first/second line)	
Supporting Actors	IT user	
Goal	Solve a problem.	
Precondition	[Workspace: request]	
Description	Actor	System
	A1) VAR1) the actor takes on an open request from his/her line. [Exception: There are no open requests] VAR2) the actor receives a request forwarded to him/her.	S1) If VAR1) The system records the actor as owner of the request. [System function: take on request]
	A2) [optional *] The actor adds information [Include UC Clarify request]	
	A3) [optional X] The actor forwards the request. VAR1) forward to second line VAR2) forward to specific expert (no matter what line) [include UC Handle request]	S3) The system forwards the request [System function: forward request] If VAR1) The system changes the owner to "not set" and the status to "second line" If VAR2: The system sets the expert as the owner and notifies him/her about the forwarded request. If he/she is logged in at the moment, the system sends an alert in a way designed to attract his/her attention (e.g. a pop-up window). Else it sends the alert as soon as the expert logs in.
	A4) [optional *] The actor looks up information of the request.	S4) The system provides information about the request. [System function: show request details]
	A5) [optional X] The actor solves the problem and closes the request.	S5) The system closes the request. The system sends the user a notification. [System function: close request]
Exceptions	[There are no open requests]: The system contains no open requests.	
Rules	None	
Quality Requirements	None	
Data, Functions	System functions: <ul style="list-style-type: none"> - take on request - show request details - add information to description field - forward request - close request 	
Postconditions	The request is closed in the system.	
Included Use Cases	Clarify request; Handle request	

Table 4: Use Case „Handle Request“

Name	Set support level	
Actor	First line supporter or second line supporter	
Supporting Actors	None	
Goal	Work on desired type of problems	
Precondition	[Workspace: supporters, request]	
Description	Actor	System
	A1) The actor requests a new type VAR1) The actor has no type assigned yet VAR2) The actor has a type assigned and wants to switch to another type	S1.1) The system checks whether there are enough supporters of the non-target type [R1][Exception: not enough supporters of the non-target type] S1.2) if the actor owns open requests, the system sets their status to "second line" [System function: update request data] S1.3) the system sets the actor's supporter type [System function: change supporter type] S1.4) the system sends the actor an acknowledgement S 1.5) the system notifies the other supporters that a change has taken place
	A2) Actor starts working as a supporter of the new type.	
	A3) Actor logs out	S3.1) System changes the actor's type to "not available" S3.2) [optional] If too few supporters of the actor's type remain logged in, system warns the other supporters
Exceptions	[not enough supporters of the non-target type] <ul style="list-style-type: none"> - If actor already has a type: the system warns the actor that a change of his/her role is not possible at the moment - If actor has no type assigned yet: the system tells the actor there are not enough supporters of the non-target type and asks the actor to work as the type needed. Actor decides to maintain his/her decision or to take the target type offered by the system. In both cases, the scenario proceeds to step S1.2. 	
Rules	R1: When the actor changes his/her type, there should be at least one other supporter who belongs to the changing actor's initial type.	
Data, Functions	Data <ul style="list-style-type: none"> - Supporter type System functions <ul style="list-style-type: none"> - change supporter type - update request data 	
Postconditions	The actor belongs to the other type	
Included Use Cases	None	

Table 5: Use Case „Set Support Level“

Name	Get statistics	
Actor	Manager	
Supporting Actors	None	
Goal	Receive statistics	
Precondition	[Workspace: Statistics]	
Description	Actor	System
	A1) The actor requests statistics for specified parameters	S1) The system offers statistics [System functions: show statistics]
	A2) [optional *] The actor triggers the statistics export in a specified format	S2) The system exports statistics in desired format [System function: export data] [Exception: No information exists for the specified parameters]
Exceptions	[Exception: No information exists for the specified parameters] • System warns actor	
Rules	None	
Data, Functions	System functions: - Show statistics - Export data	
Postconditions	no changes in system state	
Included Use Cases	None	

Table 6: Use Case „Get Statistics“

Name	Warn about orphaned requests
Actor	System
Supporting Actors	None
Goal	Warn the supporters about requests which stay open too long.
Precondition	None.
Description	System
	S1) The system checks regularly [R1] whether there are orphaned requests. S2) VAR1) If there are orphaned requests, the system reminds the owners of the orphaned requests to handle it. [System function: warn owner] VAR2) If there are no orphaned requests, the systems sends no warnings
Exceptions	
Rules	R1: If a request has been open for more than a week without having the status “parked”, it is considered orphaned.
Quality Requirements	None
Data, Functions	System functions - warn owner
Postconditions	Supporters have been warned about not completed requests.
Included Use Cases	None

Table 7: System Activity „Warn about orphaned requests“

7. Workspaces

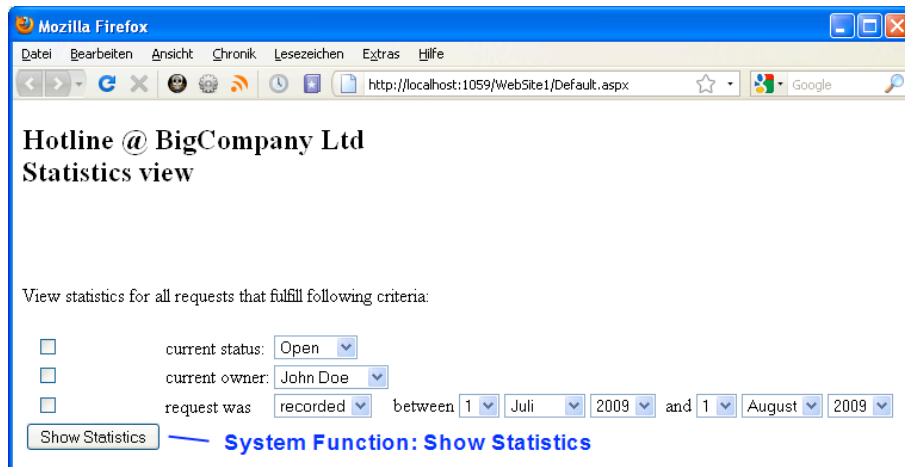


Figure 3: Workspace statistics, screen 1

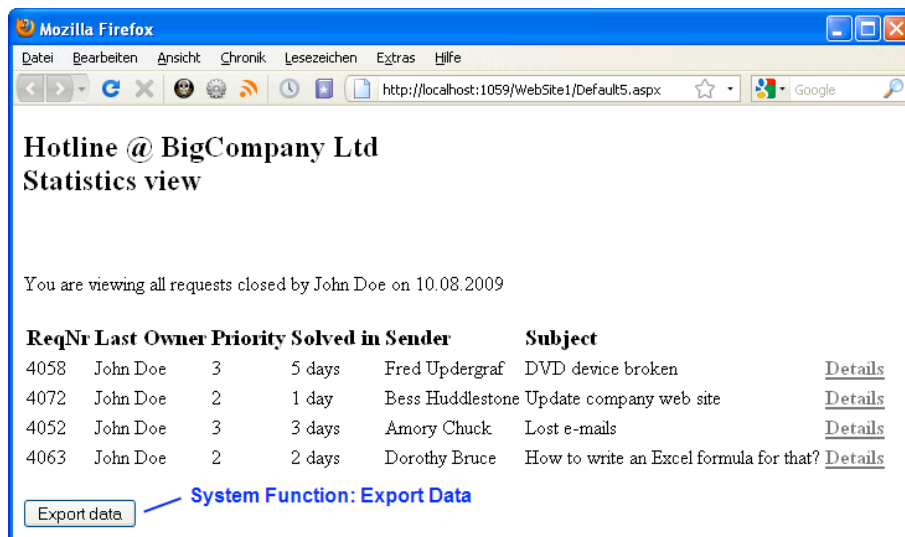


Figure 4: Workspace statistics, screen 2

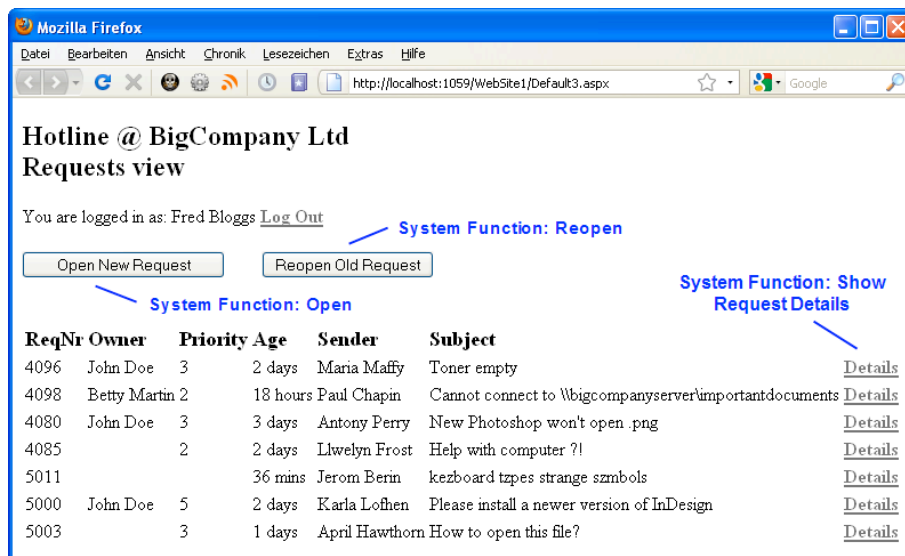


Figure 5: Workspace request, screen 1



Figure 6: Workspace request, screen 2

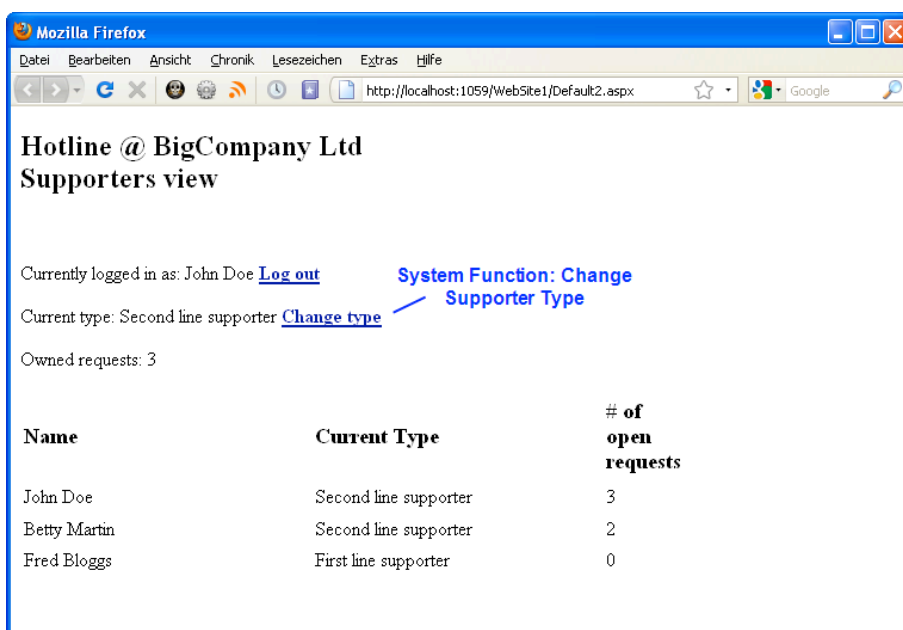


Figure 7: Workspace supporters