Systementwicklungsprojekt

MunichHunt

A location based multiplayer game

Mathias Kellerer
Supervisor: Björn Schwerdtfeger
Director: Gudrun Klinker

Fachgebiet Augmented Reality, Technische Universität München,
Boltzmannstrasse 3, Garching b. München, Germany
kellerem@in.tum.de
http://campar.in.tum.de

October 29, 2007

Contents

1 Game Concept **3**

2 Current Game Rules **4**

3 Related Work **4**

4 Realisation **5**
4.1 Hardware 5
4.2 Software 6
4.3 Application 6
4.3.1 Gameboard Implementation 7
4.3.2 Client User Interface 8
4.3.3 Communication 10
4.3.4 Synchronisation 10
4.3.5 Tracking 10
4.3.6 Game Server / Map Editor 11
4.4 Cooperation with Vodafone 11
4.5 Test Sessions 12

5 Conclusion **13**

A Appendix **15**

Abstract

A cops and robbers game has been developed during this "Systementwicklungsprojekt", where real players alias cops move around "Munich" in order to catch one real player (the robber). All players are equipped with mobile phones running the game application to support the players with appropriate game information. In this document the game idea, which is based on a popular board game called "Scotland Yard", and game rules are elucidated just as the components forming the game application. Moreover some test sessions have been carried out, which will also be presented.

1 Game Concept

"MunichHunt" is a location based multiplayer game for playing cops and robbers in Munich (or another city or only a dedicated area) and is based on the board game "Scotland Yard", by Ravensburger AG, Germany.
The cops hunt one robber which is called "Mr. X" by moving around the city. Therefore only dedicated ways are allowed, which are specified by the game application and are from different kinds: subway, underground, bus and walking distances. Public transportation is limited to a subnet in order to reduce the complexity of possible moving directions. For every travelled distance the players have to give appropriate tokens out of a limited number.
While moving around the location of the players is tracked manually. This means that every player has to provide his location via the application by selecting the station where he is located next to. Additional tracking could be done in conjunction with other systems like GPS, triangulation and so on, but is not implemented, yet. The position data (and other data) is stored by the game application and is periodically exchanged among all players by sending them to a server. The server collects the data provided by every player and distributes them to the other players.
After receiving new game data (position updates for other players, messages, ...) the new data is shown to the player by updating the visualisation. The visualisation consists of a city map, a network plan (= dedicated ways), player positions and an interface for interacting with the application.
One special case regards the visibility of "Mr. X" because his current position is not always shown to the other players. The shown position of "Mr. X" is only updated to the current one, after "Mr. X" has travelled a predefined number of distances. Therefore the chasers have to guess the actual position. After every "invisible" move of "Mr. X" a hint is given, because it would be to difficult to guess his current position. The hint consists only of the transportation type used by "Mr. X".
For communication purposes, e. g. arranging chaser moves, a messaging service is available where text messages can be sent to one or all players. In further implementations other ways of communication are imaginable.
The game is over whether one player has caught "Mr. X" or all players are running out of tokens. If this is the case the players are no longer able to move around in order to catch the robber. Basically "Mr. X" will be marked as caught, if someone arrives at the same station where "Mr. X" resides on. Moreover different catching possibilities are imaginable in order to overcome limitations of the game application. Players, who are no longer participating are marked as inactive and are no longer considered.

All players have to be equipped with a Java enabled mobile phone which provides an operational data connection so that the game data can be exchanged. There is no reason for expensive and fast communication technologies (like UMTS) because of the use of a small communication protocol and the fact that the game needs no "real time communication".

2 Current Game Rules

Every game consists of a set of rules. Likewise "MunichHunt" has a few rules build in. These are check periodically and if a violation occurs, appropriate actions will be triggered.
Currently only basic game rules are implemented. Firstly the application checks whether "Mr. X" has been caught by one chaser or not. A "catch" happens if one player resides on the same station as "Mr. X" resides on. Instead a time window is defined and the "catch" can only happen inside the time window. This represents a minor issue, because after the time window is over "Mr. X" can not be caught anymore by the chasers. In order to avoid this circumstance it is imaginable to force every player (especially "Mr. X") to leave a station within a specific period of time, which could be adjusted dynamically depending on the used means of travel. Moreover it could be combined with an automatic tracking method (see chapter 4.3.5) to determine whether a player has left a station or not. Secondly every players movability is tested since without tickets players can not move any more in order to catch "Mr. X". If all chasers were no longer able to move around "Mr. X" would win the game. Thirdly it is checked if any player has left the game before it ends. This is done via an inactive flag. The flag will be set for the according player if his last timestamp from one station is older than a predefined value. For the test sessions this value was set to 20 minutes. After the flag has been set the "inactive player" will not be displayed any more. If all chasers are marked as "inactive" "Mr. X" will win the match and the other way around the chasers will win if "Mr. X" is no longer active.
Because it would be to easy to catch "Mr. X" his current position is not shown to the chasers the whole time. His position is only revealed after a predefined number of moved stations. To give the chasers a hint where "Mr. X" could be, only the used ticket type (the used means of transportation) is displayed to the chasers.

3 Related Work

The idea of supporting cops and robbers like games in the city by ubiquitous computing devices is not completely novel. Can You See Me Now [1] is a game of catch but with a twist. Online players are chased through a virtual model of a city by runners or street players, who have to traverse the actual city streets in order to capture the online players. Up to fifteen members of the public at a time can be online players, accessing the virtual city model over the Internet. The street players are equipped with PDAs, GPS and walkie-talkies.
Catch Me If You Can [2] is a proposal of an outdoor "Scotland Yard" like game using SMS Communication. Live Action Super Scotland Yard [3] is a another

way of playing “Scotland Yard” outdoor. However the game is mainly managed by a headquarter which is called from time to time by the players via mobile phones.
Adopted from [4], paragraph 3.

4 Realisation

The system architecture of “MunichHunt” is shown in figure 1. It is divided into several clients (mobile phones) and one server (standard PC) together forming a client/server architecture. The players are equipped with mobile phones and move around the city while the server saves the generated game data from the clients and distributes them among the other clients.

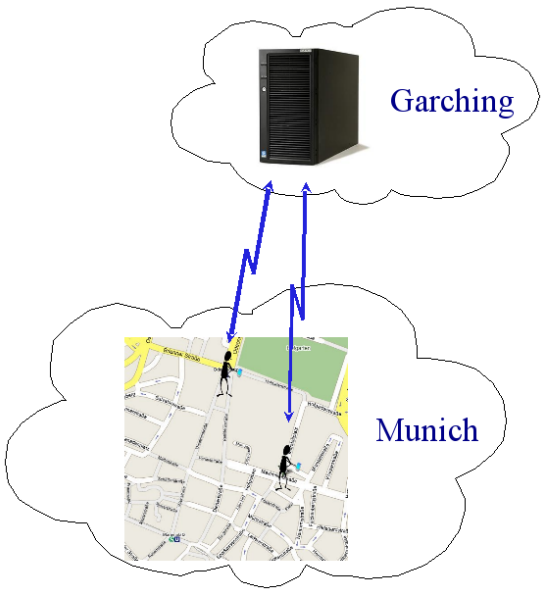


Figure 1: System overview

In order to realise games or applications on mobile phones some extra tools are needed to build, deploy, run, test and debug. Also it is very interesting how to set up a game so that it is well balanced and the players were not bored after a while. In the following it will be discussed how “MunichHunt” is designed. Furthermore the cooperation with “Vodafone Group R&D” raised some aspects/work which will also be mentioned.

4.1 Hardware

During this project “Nokia 6280” mobile phones[1] were used. Because of a “Symbian Series 40” operating system only Java (“J2ME”) binaries can be deployed and executed. But this has the advantage, that the application should

[1] Device specific features can be found here: `http://www.forum.nokia.com/devices/6280`, 13 September, 2007

run on other mobile phones which support the same language features. The only requirements for the used mobile phones are:

- Java enabled (CLDC 1.1, MIDP 2.0 and JSR 75)
- Data connection (GPRS or faster)
- Display resolution: 240*320 pixel

Currently "MunichHunt" is only tested on the "Nokia 6280" phones and therefore no statement can be given, whether the application runs on other phones. For more device details and documents regarding application development for "Nokia" phones, please refer to the development website of "Nokia"[2].

4.2 Software

For application development the IDE "Eclipse"[3] and the Nokia developer suite (for J2ME) "Carbide.j"[4] has been used. Carbide.j is an Eclipse plugin for building the "jar", that can be executed on the mobile phones. Nokia decided to stop developing Carbide.j due to open source alternatives, especially "EclipseME"[5]. To deploy the application sending the "jar" via bluetooth to the device is enough. If bluetooth is not available, the "Nokia PC Suite" can also be used to copy the file to the phone. After deployment one security setting has to be changed. More precisely: Access to the filesystem of the phone must be allowed to show the map.
In the "Series 40 Platform SDKs" a specific simulator for Nokia phones is provided. The simulator can be used in order to test and debug the application in an appropriate way. The SDK can be found and downloaded from the Nokia homepage[6].

4.3 Application

The "MunichHunt" game application is separated into clients and a server composed of several blocks with different functionality. More precisely this means that the client application is made up of components which are responsible for showing the city map (see section 4.3.1) and the client user interface (see chapter 4.3.2) on the screen of the mobile phone. The tracking component (see chapter 4.3.5) provides methods for "position estimation" of the player. Violation of game rules and the state of the game are checked by the game rule component (see chapter 2). Furthermore components for data exchange (see chapter 4.3.3) and data synchronisation (see chapter 4.3.4) with a game server are present.
Finally a game server component (see chapter 4.3.6), which is executed on a normal PC, acts as a server. In the following this components will be presented.

---

[2] `http://www.forum.nokia.com`, 13 September, 2007
[3] `http://www.eclipse.org/`, 13 September, 2007
[4] `http://www.forum.nokia.com/main/resources/tools_and_sdks/carbide/what_has_happened.html`, 13 September, 2007
[5] `http://www.eclipseme.org/`, 13 September, 2007
[6] `http://www.forum.nokia.com/info/sw.nokia.com/id/cc48f9a1-f5cf-447b-bdba-c4d41b3d05ce/Series_40_Platform_SDKs.html`, 13 September, 2007

4.3.1 Gameboard Implementation

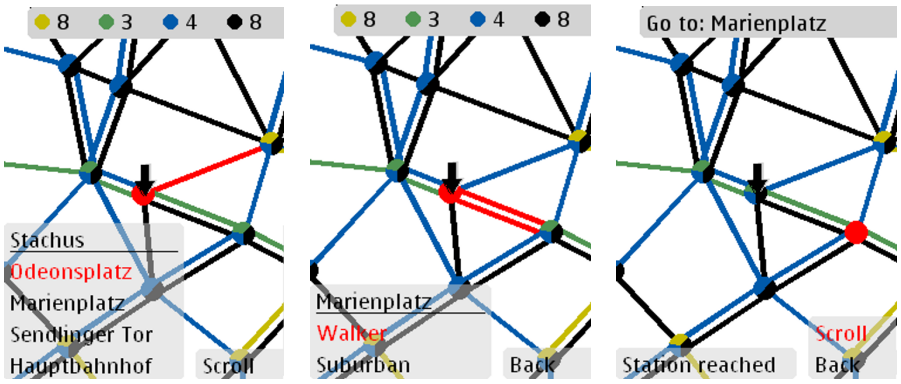
Map visualisation consists of two parts. Firstly a normal city map as background image is shown to the player so that he can easily navigate through an area or city which he is not familiar with. Secondly a network plan, which shows the allowed public transportation, is overlaid over the city map. Figure 2 shows a small cutout of the whole map used in "MunichHunt".

Figure 2: Map and network plan

Due to limitations of the used mobile phone (maximal heap size is 2 Mbyte) it is not possible to use one map file for the whole map. Therefore the whole map is split up into quadratic map tiles. These tiles are dynamically loaded into memory. One challenge is that the size of the tiles may not be to large, since loading large tiles slow down the performance of the device. Moreover it is not possible to store the tiles in the jar, because the maximal jar file size is limited to 500 kbyte (for the used device). But the "FileConnection" API (JSR 75) is used to get access to the filesystem of the mobile phone in order to store the tiles (in one file).
The used network plan is represented as an undirected graph. It is composed of vertices representing the stations of the public transportation and edges. The edges represent the allowed means of transportation for travelling from one station to the neighbour station. On the screen of the mobile phone vertices are visualised as circles and edges as lines. Circles and lines are colourised with the appropriate colour of the representing public transportation. In order to initialise the structure of the graph with data, a file ("networkPlan.txt") with all necessary information is needed and is included in the jar. To easily adjust the topology of the graph, a graphical editor has been written. For more information about the editor please refer to section 4.3.6.

4.3.2 Client User Interface

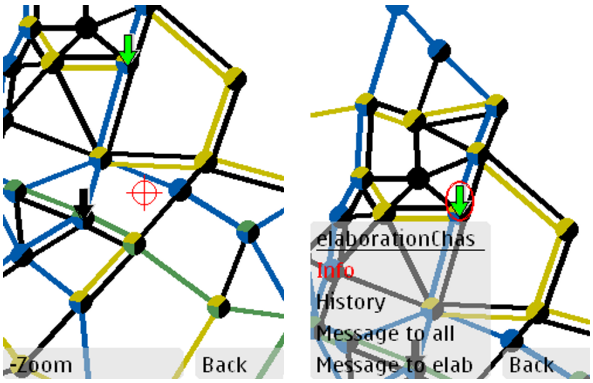
Conceptually the client user interface is separated into two parts, in order to make it easier for the players to handle the game actions appropriate. On the one hand it can be used for navigating on the network plan and on the other hand for scrolling around the map and getting information about other players as well as sending messages.
Figure 3 shows the flow of events when a user wants to go from one station to a neighbour station. If the user interface is in "ScrollMode", the player has to switch back to "NetworkPlanMode", where the current station of the player is highlighted (shown in "red") and automatically centred on the screen. In the bottom left corner a menu with all possible neighbour stations is displayed and one path to a neighbour station is highlighted on the network plan as well as the according menu entry (see figure 3(a)). After selecting a neighbour station the "station selection" is replaced by a menu for selecting the means of travel which can be used (see figure 3(b)). Finally the arrival at the station must be confirmed which is shown in figure 3(c).
Selecting an entry in the menu is done by using the "UP" and "DOWN" keys of the mobile phones navigation pad. Moreover confirming a choice is done by pressing the "LEFT" or "RIGHT" softkey under the appropriate selection. During "NetworkPlanMode" the menu is not fully opaque but up to a significant level transparent, so that the user can recognise the stations and their connections which are covered by the shown menu.



(a) Selecting next station (b) Selecting means of travel (c) Confirm station reached

Figure 3: UI - "NetworkPlanMode"

In "ScrollMode" it is possible to scroll around the map to find other players, the last known position of "Mr. X" and so on. A crosshair is shown in the middle of the screen for selecting players. If another player has been selected (see figure 4(b)), information about the player and his travel history can be seen as well as sending messages to all or the selected player is possible. The travel history includes every station (with a timestamp) and the used ticket, which has been used by the player to travel to the station.
In order to display information, like histories or messages from other players, a facility to display this accordingly is needed. Therefore an opaque panel is used, which dynamically adjusts to the size of the content. It is called "InfoScreen".



(a) Scrolling around the map (b) Select a player

Figure 4: UI - "ScrollMode"

Because the panel is not larger than the display the content can be scrolled up and down if it is not possible to display all at once. To indicate the scroll ability two arrows are shown on the right side of the panel (see figure 5, only down arrow is shown in the picture).

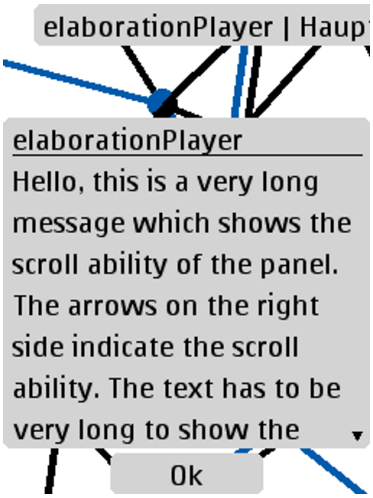


Figure 5: UI - Panel with scroll abilitiy

Moreover to display permanent information to the user, a so called "InfoBar" is shown in the display. It can only be used for minor important information, because the user is not forced to confirm the shown information (as this is the case for an "InfoScreen"). The "InfoBar" can be seen in figure 3 on top of the screens. In the first two pictures the ticket status is shown through the "InfoBar" and an advise to go to a specific station in the third one.
Additionally it is possible to zoom in and out of the map in discrete steps (> 10). Because it is not possible to store map tiles for all discrete zooming steps only map tiles for the maximum zoom factor are provided. If the zoom factor is not maximal the map will be removed and only the network plan will be displayed.

The zoom factor is adjusted automatically in "NetworkPlanMode" in order to show all neighbour stations so that the player gets an overview of all travelling possibilities. To support the player in getting a quick overview of the game state shortcuts to three zoom factors (min, medium, max) are present.

4.3.3 Communication

During gameplay status information, such as player positions, messages and so on, will be exchanged. Therefore a dedicated communication mechanism is needed. As mobile phones are used as platform it is obvious to use the communication technology provided by the mobile network. This means that data exchange is done via TCP/IP (in this application over GPRS), which provides a reliable communication channel and is also supported by J2ME (see "Connector" class[7]).
In order to distinguish between different data packages an application specific package format has been used. Every package contains a "message ID", which specifies the type of the data, and the actual "payload". A "message dispatcher" dispatches the received data according to the "message ID" and delegates them to the appropriate facilities.

4.3.4 Synchronisation

Another question, that arises, is the question how to synchronise the player data among each other. The approach, which has been chosen is a "best effort" algorithm. It means, that every client tries in regular intervals to exchange his game data (which includes the positions of the player and the appropriate time stamps since the last synchronisation) with the server (see chapter 4.3.6) and to receive the game data from the other players. This implies that the game status is not older than the length of the synchronisation interval, except for players who are suffering a dead spot (i. e. subway). Therefore players without network coverage are highly motivated to be back "online" in order to update the positions of the other players (especially "Mr. X") as soon as possible.
But this fact can also be used to travel unseen but as well can not be seen by the other players. It is imaginable to travel several stations by subway without network coverage. This results in several minutes where no data update can be sent and retrieved. In order to avoid such "blind travels" the game rules or the network plan could be adjusted appropriately which means that longer travels without network coverage are innately avoided.

4.3.5 Tracking

Another important part of "MunichHunt" is the localisation of the players. Primary the position is given by the player manually, as mentioned in section 4.3.2, by selecting the stations during gameplay. Therefore no additional effort is necessary but makes it very easy for players to use creative gameplay techniques[8]. In combination with other tracking methods like "GPS" or "Triangulation" this can be prohibited. The manually entered position information can be fusioned

[7] `http://java.sun.com/javame/reference/apis/jsr118/javax/microedition/io/Connector.html`, 22 September, 2007
[8] Some people would say "cheating"

with the automatically provided positions. A drawback is that both mentioned methods also need "coverage". One possible solution, to solve the "coverage" problem, is to get rid of underground connections. But without underground connections maybe an important part will be missing. It is also possible to deal with the problems arisen with underground connections. For example game rules could be adjusted appropriately, to force players to achieve "coverage" again after travelling with underground transportation. Alternatively players could gain extra points or extra tickets if they successfully avoided a loss of "coverage" during gameplay.

4.3.6 Game Server / Map Editor

As previously mentioned, it is not always possible for the players to gain coverage from mobile network, especially when travelling by subway. This leads to the question how to exchange data with other players who are currently not reachable. A solution for this problem is to use a client/server architecture. In contrast to the clients, which are the mobile phones of the players, the server is a standard personal computer with a permanent internet connection. The server stores all incoming data packages and distributes them to the other clients and guarantees the delivery, because the server is always reachable for every player. In order to distribute the game date to all players, the server collects the data from every player when they are requesting updates. Received data is first checked for consistency and afterwards stored internally. If a player connects to the server the missing game data is pushed to the client and is included in the already existent datastructure.
Moreover the server application gives the opportunity to build new maps and network plans in a short way. Every image (containing a city map, ...) can be used as map and it must be possible to separate the image into equal tiles. In order to construct a network plan, only stations must be added to the map, which can be connected via edges. Stations and paths have to be at least from one type of public transportation. Figure 6 shows the view of the map and how stations are inserted.
The server can also be used to spectate the game remotely as the server knows all player data and appropriately visualises them. A replay function is currently missing but can be implemented in order to reuse existing game data. This means that after a real game session it is possible to review the game by the players to analyse their gameplay and to recognise how they were on the verge of catching "Mr. X".

4.4 Cooperation with Vodafone

Vodafone Group R&D supported this project by providing mobile phones and data traffic in order to develop the application and to conduct some test sessions (see chapter 4.5).
Furthermore "MunichHunt" was presented in the course of *Vodafone's* (1^{st}) *betavine forum*[9]. Afterwards the application has been published within the betavine platform where it can also be downloaded[10].

[9] `http://www.iom.bwl.uni-muenchen.de/files/betavine.pdf`, 29 September, 2007
[10] `http://www.vodafonebetavine.net/web/MunichHunt`

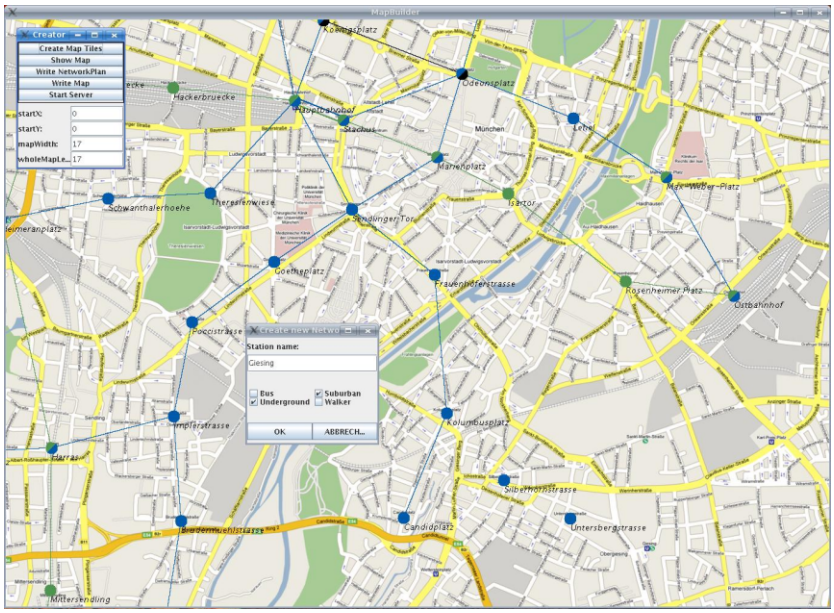


Figure 6: Game Server - Building a new map

At the beginning of the cooperation a set of "Use Cases" has been developed. Basically during game development the focus was to achieve a running application in the ballmark for a "Systementwicklungsprojekt". Therefore more use cases have been considered and described than are currently implemented.
The use cases can be found in the appendix A.

4.5 Test Sessions

After the main part of the implementation was finished, the application has been tested and debugged until two test sessions took place in the city of Munich. Before the two test sessions took place some "virtual sessions" were scheduled. "Virtual session" means that one or two people control more than one player (so called "virtual players") and do not move around the city. So they can arrange the moves of the "virtual players" in order to find bugs in the application. On the one hand the application with "virtual players" runs on real hardware and on the other hand can reside on a mobile phone simulator, which provides appropriate debugging output. The concept of "virtual players" is also imaginable for the real game. They can coexist beside real players, but must not be located to the city of Munich or the place were the game takes place. In order to avoid that "virtual players" move more quickly than real players it is necessary to constrain the time to travel from one vertice on the network plan to a neighbouring one. Moreover an option to resume a game session, after closing the application or a crash has been added, because it is very hard to claim that an application is absolute "bug free" and a resume feature improves usability of the application. At the end of the "Systementwicklungsprojekt" two test sessions with real players were conducted. The first sessions was made up of four players and lasted about one and a half hour. Finally "Mr. X" has been caught by one player.

It is remarkable that "Mr. X' and the chasers spent a lot of tickets for subway connections. This preference is due to many subway tickets, the possibility to travel a great distance in a short time and the frequent interval of trains compared to the other transportation opportunities (especially compared with bus/walking connections). To avoid extreme use of subway connections and to give the players reasons to use other means of transportation, the amount of tickets and the network plan had been adjusted for the second test.
The second session, with six people, splits up in three short sessions because "Mr. X" has always been caught very fast. Afterwards an evaluation of the session has been done independently from this "Systementwicklungsprojekt", but a few aspects will be presented in the following. The players mentioned that typing messages during the game is not great fun. A better alternative would be the use of voice messages, because a short voice message can be recorded much faster compared to a message, which has to be typed. Moreover the players complained about the distribution of players at the beginning of the game. The intention was that players start at a position next to her real position. If more players meet together at the beginning of a game the distribution is not very good. Therefore a random distribution for all players is necessary. Also a point of criticism was the balance of the network plan, because more alternative connections were missing.

5 Conclusion

A nice game for playing cops and robbers in the city of Munich has been developed. Some players are hunting one player through the city using a subnet of the available public transportation. They are all equipped with mobile phones running the game application. The application collects the position data of every player and synchronises the game data with a server, shows a city map, network plan and players on one map, provides a messaging service and so on. If the robber is catched the game will be over.
Regarding the test sessions further "fine tuning" should be done in order to balance the game more. This includes the amount of tickets in conjunction with the network plan. Adjusting the balance will result in more test sessions. Moreover some features like "voice messages" can be integrated and makes playing easier for the players. But this is behind the focus of this "Systementwicklungsprojekt".
Acknowledgement: Many thanks to Vodafone Group R&D who supported the project with mobile phones, data traffic and inspiring ideas. Moreover many thanks to the people who helped us hunting "Mr. X" during the test sessions and providing feedback afterwards.

List of Figures

1 System overview 5
2 Map and network plan 7
3 UI - “NetworkPlanMode” 8
4 UI - “ScrollMode” 9
5 UI - Panel with scroll abilitiy 9
6 Game Server - Building a new map 12

References

[1] Steve Benford, Andy Crabtree, Martin Flintham, Adam Drozd, Rob Anastasi, Mark Paxton, Nick Tandavanitj, Matt Adams, and Ju Row-Farr. Can you see me now? *ACM Trans. Comput.-Hum. Interact.*, 13(1):100–133, 2006.

[2] Zain M. Chandan and Anurag A. Laddha. “catch me if you can” – a mobile p2p application. `http://www-static.cc.gatech.edu/grads/z/zainmc/papers/mobilep2p.pdf`, 13 September, 2007.

[3] Joel Friesen. Live action super scotland yard. `http://www.culturehole.com/article.asp?blog_id=14`, 13 September, 2007.

[4] Björn Schwerdtfeger, Mathias Kellerer, and Gudrun Klinker. Munich hunt: Playing cops and robbers all over munich. In *Adjunct Proc. of 9th International Conference on Ubiquitous Computing (UbiComp07)*, September 2007.

A Appendix

Following pages contain the Use Cases of "MunichHunt":

Table of Contents

1. Actors....4
1.1. Admin....4
1.2. Server....4
1.3. Client....4
1.4. Community User....4
1.5. New Player....4
1.6. Player....4
1.7. Chaser....4
1.8. Mr. X....5
1.9. Spectator....5
2. Use case diagram....6
3. Use cases – Admin related....8
3.1. Start server application....8
3.2. Shutdown server application....8
4. Use cases – Server related....9
4.1. Start game session....9
4.2. Check game status....9
4.3. Synchronise with clients....10
4.4. Ban players....10
4.5. End game session....11
5. Use cases – Client related....12
5.1. Connect to server....12
5.2. Synchronise with server....12
5.3. Locate position....13
5.4. Show game data to player....14
6. Use cases – Community User related....15
6.1. Create community account....15
6.2. Login to community....15
6.3. Logout from community....16
7. Use cases – New Player related....17
7.1. Join game....17
7.2. Choose role....17
8. Use cases – Player related....19
8.1. Move around....19
8.2. Communicate with other players....19
8.3. Leave game....20
9. Use cases – Chaser related....21
9.1. Catch Mr. X....21
9.2. Identify Mr. X....21
10. Use cases – Spectator related....23
10.1. Spectate at a game....23
11. Annotations....24

Document History

Author	*Date*	*Description*
Mathias Kellerer	12/12/06	• v 0.1, initial version

1. Actors

1.1. Admin

The actor Admin takes care about the server and is responsible for the Munich Hunt application.

1.2. Server

The main purpose of the Server is to hold the game logic and to provide informations about the players (e.g. positions, ticket history, and so on) to all other players.

1.3. Client

The Client describes the hardware which is used by the players. They communicate with the server and present the game informations to the players.

1.4. Community User

The actor Community User is the overall instance of a person who interacts with the Munich Hunt system.
(The community part of the system will not be implemented yet.)

1.5. New Player

Someone who intends to play Munich Hunt.

1.6. Player

The actor Player is an abstraction of the actors Chaser and Mr. X.

1.7. Chaser

A Chaser is a person who acts as a member of the “police” and tries to catch Mr. X by moving around Munich.

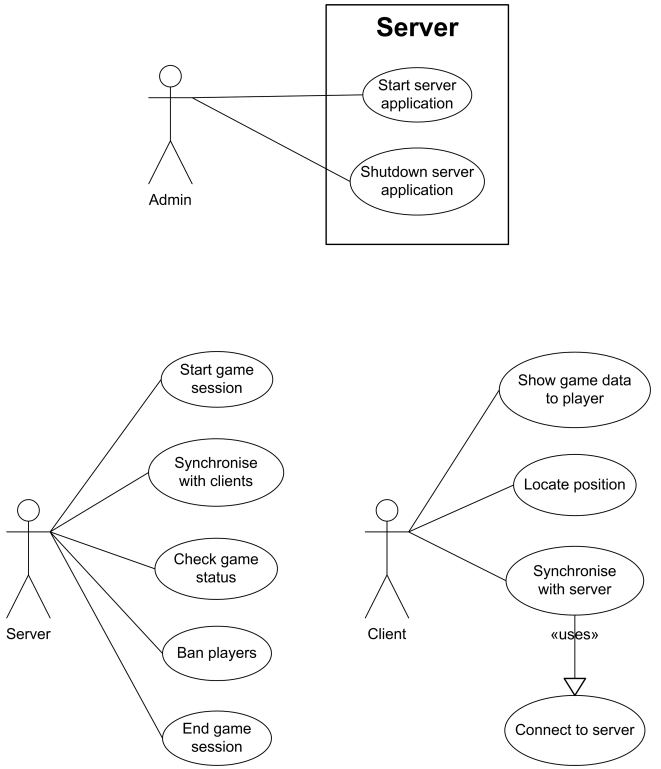
1.8. Mr. X

Mr. X is one player who tries to avoid to be caught by the Chasers.

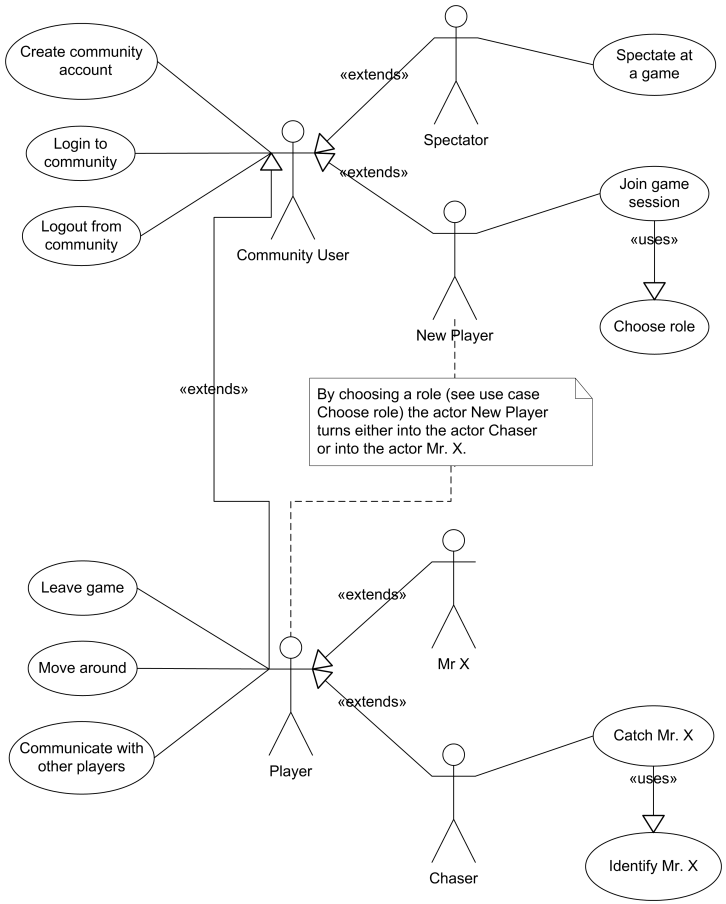
1.9. Spectator

A person who wants to spectate at a currently running Munich Hunt game session.

2. Use case diagram



Use case diagram – Munich Hunt		
Mathias Kellerer	12/12/06	v 0.1



Use case diagram – Munich Hunt		
Mathias Kellerer	12/12/06	v 0.1

3. Use cases – Admin related

3.1. Start server application

Use case name	Start server application
Description	The server application for Munich Hunt is initialised. After initialisation it is possible to start game sessions (server could host several sessions) and to spectate at already started.
Actors	Admin
Includes	
Triggers	
Preconditions	
Postconditions	The server is now ready for hosting Munich Hunt game sessions.
Flow of events	
Alternative paths	
Exceptions	
Notes	
Author and date	Mathias Kellerer, 12/12/06
Use case history	• v 0.1, initial version

3.2. Shutdown server application

Use case name	Shutdown server application
Description	The server application terminates.
Actors	Admin
Includes	
Triggers	
Preconditions	No more game sessions should run on the server.
Postconditions	It is not possible to play Munich Hunt any longer.
Flow of events	
Alternative paths	
Exceptions	
Notes	
Author and date	Mathias Kellerer, 12/12/06
Use case history	• v 0.1, initial version

4. Use cases – Server related

4.1. Start game session

Use case name	Start game session
Description	After enough players joined the server (see 7.1 Join game) a game session is started and the game begins.
Actors	Server
Includes	
Triggers	Joining of a new player (see 7.1 Join game).
Preconditions	The server application has been started before (see 3.1 Start server application) and following conditions are fulfilled: ● Number of players representing Mr. X is greater than 0 ● Number of players representing Chasers is greater than 0
Postconditions	A new game session of Munich Hunt is running on the server.
Flow of events	1. A new player connects to the server (see 7.1 Join game) 2. Conditions (see preconditions) are checked 3. If enough players are connected to the server a session will be started
Alternative paths	
Exceptions	Not enough players are currently connected to the server and therefore the session will not be started.
Notes	
Author and date	Mathias Kellerer, 12/12/06
Use case history	● v 0.1, initial version

4.2. Check game status

Use case name	Check game status
Description	The server checks after every synchronisation with one client compliance with the rules and determines the game status. That means it is checked whether game is over (see 4.5 End game session), time is over or some special conditions are fulfilled (see notes).
Actors	Server
Includes	
Triggers	Triggered by the client use case 5.2 Synchronise with server.
Preconditions	
Postconditions	
Flow of events	
Alternative paths	
Exceptions	Violated rules, therefore a player will be banned (see 4.4 Ban players).

Use case name	Check game status
Notes	The term “time is over” should be defined. In “Scotland Yard” the term “time is over” means that Mr. X could not be caught for a defined number of moves. It is possible that some special conditions could also be checked. For example a player is near to a point with informations which should shown to the player (e.g. advertisements, ...).
Author and date	Mathias Kellerer, 12/12/06
Use case history	● v 0.1, initial version

4.3. Synchronise with clients

Use case name	Synchronise with clients
Description	The server sends all necessary informations (positions and messages of other players, ticket history, ...) to the client which currently synchronises with the server (see 5.2 Synchronise with server).
Actors	Server
Includes	
Triggers	A client wants to synchronise with the server (see 5.2 Synchronise with server).
Preconditions	All necessary data should be collected from the other clients in a defined time slice.
Postconditions	All relevant informations updates have been sent to the client.
Flow of events	
Alternative paths	
Exceptions	Synchronisation could be interrupted or one or more clients have not synchronised their data (see notes).
Notes	For the case that clients have not synchronised their data within a defined time they could be banned from the currently running game (see 4.4 Ban players).
Author and date	Mathias Kellerer, 12/12/06
Use case history	● v 0.1, initial version

4.4. Ban players

Use case name	Ban players
Description	It is possible that the server can exclude players from a game session.
Actors	Server
Includes	
Triggers	A player seems to cheat or does not to take part in the game any more.

Use case name	Ban players
Preconditions	Player is a registered player and takes part in the game.
Postconditions	Player has been banned from the game and does not take part any more
Flow of events	
Alternative paths	
Exceptions	
Notes	Players should be banned carefully. It could be possible that they are only suffering dead spots.
Author and date	Mathias Kellerer, 12/12/06
Use case history	• v 0.1, initial version

4.5. End game session

Use case name	End game session
Description	The game session is over either Mr. X has been caught (see 9.1 Catch Mr. X) or the time is over. The server informs all players that the game is over now and ends the game session appropriate.
Actors	Server
Includes	
Triggers	Triggered by use case 4.2 Check game status.
Preconditions	Game is running and Mr. X has not been caught before.
Postconditions	Game is over and the session does not exist any more.
Flow of events	
Alternative paths	
Exceptions	
Notes	
Author and date	Mathias Kellerer, 12/12/06
Use case history	• v 0.1, initial version

5. Use cases – Client related

5.1. Connect to server

Use case name	Connect to server
Description	For communication issues (mostly for synchronisation with the server, see 5.2 Synchronise with server) a dedicated connection to the internet is needed. The connection is set up on demand using an appropriate and local available communication channel. After finishing the communication the connection will be disconnected.
Actors	Client
Includes	
Triggers	Triggered by use case 5.2 Synchronise with server.
Preconditions	An appropriate communication channel is available: • UMTS • WLAN • ...
Postconditions	Communication has been done.
Flow of events	
Alternative paths	
Exceptions	No communication channel is available.
Notes	
Author and date	Mathias Kellerer, 12/12/06
Use case history	• v 0.1, initial version

5.2. Synchronise with server

Use case name	Synchronise with server
Description	All data sets collected (by the clients) during the game has to be published to the other clients after a defined period of time (according to synchronisation policies). Therefore a connection to the internet is set up by every client (see 5.1 Connect to server) and the recorded data set of the client (recorded positioning data, used tickets, ...) are sent to the server. The client receives from the server the data sets from the other clients (see 4.3 Synchronise with clients).
Actors	Clients
Includes	
Triggers	Triggered by synchronisation policy: • Periodical • Permanent • Best effort • Need (e.g. see 9.1 Catch Mr. X or 8.2 Communicate with other players) • ...

Use case name	Synchronise with server
Preconditions	
Postconditions	The client knows all other client data sets (positions, movements, coordination actions, …) up to a specific time shift and the server is now aware of all actions taken by the client who has synchronised his data sets.
Flow of events	
Alternative paths	
Exceptions	
Notes	
Author and date	Mathias Kellerer, 12/12/06
Use case history	● v 0.1, initial version

5.3. Locate position

Use case name	Locate position
Description	The position of every player has to be determined and saved.
Actors	Client
Includes	
Triggers	
Preconditions	Current position not known – player has moved.
Postconditions	Current position is known.
Flow of events	
Alternative paths	
Exceptions	Position detection not always possible (e. g. while going by subway).
Notes	In order to get the position of a player different techniques are possible: ● GPS ● Triangulation ● Player based (player tells the application at which station he currently is) ● ... The positioning should be done by a best effort strategy in order to get as much positions as possible but it is also possible to do this periodical.
Author and date	Mathias Kellerer, 12/12/06
Use case history	● v 0.1, initial version

5.4. Show game data to player

Use case name	Show game data to player
Description	Informations about available means of travel (MVV subnet), shortest way to Mr. X last known position, positions of other Chasers/Mr. X, important points where a player has to go, markers and messages from players, and so on are shown to the player by the client.
Actors	Client
Includes	
Triggers	
Preconditions	
Postconditions	
Flow of events	
Alternative paths	
Exceptions	
Notes	Different techniques could be used to visualise the data: ● Map based ● Text ● Radar like ● ... Which techniques could be used depends on the clients hardware. If more than one visualisation is possible the player could choose one.
Author and date	Mathias Kellerer, 12/12/06
Use case history	● v 0.1, initial version

6. Use cases – Community User related

6.1. Create community account

Use case name	Create community account
Description	Someone who wants to take part in the Munich Hunt community has to create a community account and enter personal informations first in order be known by the system.
Actors	Community User
Includes	
Triggers	
Preconditions	The person has no community account yet.
Postconditions	A community account has been created.
Flow of events	
Alternative paths	
Exceptions	
Notes	It is possible to require a photo of the person in order to use this photo later to identify Mr. X (see 9.2 Identify Mr. X).
Author and date	Mathias Kellerer, 12/12/06
Use case history	● v 0.1, initial version

6.2. Login to community

Use case name	Login to community
Description	Before someone can join a game (see 7.1 Join game) or spectate at a Munich Hunt session (see 10.1 Spectate at a game) the person has to login to the community.
Actors	Community User
Includes	
Triggers	
Preconditions	The Community User is not logged in.
Postconditions	Afterwards the Community User is logged in and is ready for interacting with the system.
Flow of events	
Alternative paths	
Exceptions	Wrong login informations (e.g. username or password).
Notes	
Author and date	Mathias Kellerer, 12/12/06
Use case history	● v 0.1, initial version

6.3. Logout from community

Use case name	Logout from community
Description	After a Community User has completed all sessions and wants no longer to take part in another sessions it would be recommended to logout from the community.
Actors	Community User
Includes	
Triggers	
Preconditions	The Community User is currently logged in.
Postconditions	Afterwards the Community User is no longer logged in.
Flow of events	
Alternative paths	
Exceptions	Community Users did not properly sign off. After a timeout the Community User will automatically be logged off.
Notes	
Author and date	Mathias Kellerer, 12/12/06
Use case history	● v 0.1, initial version

7. Use cases – New Player related

7.1. Join game

Use case name	Join game
Description	In order to take part in a Munich Hunt game everybody has to join to a game session. This session could already be started or waiting for enough players to start (see 4.1 Start game session).
Actors	New player
Includes	Choose role
Triggers	
Preconditions	Player does not participate in another Munich Hunt session and has not spectated at the game for a defined period of time (see 10.1 Spectate at a game).
Postconditions	Player has joined to the selected Munich Hunt session.
Flow of events	1. Client initiates a connection to the server (see 5.1 Connect to server) 2. Advertise intention to take part 3. Choose side (see 7.2 Choose role) 4. Server generates start position (could be the current position) 5. New player moves to the starting position 6. After arrival the game start for the new player if session has been started before. Otherwise the player has to wait until the session has been started (see 4.1 Start game session).
Alternative paths	
Exceptions	Player does not choose a role and therefore the player will be assigned.
Notes	
Author and date	Mathias Kellerer, 12/12/06
Use case history	● v 0.1, initial version

7.2. Choose role

Use case name	Choose role
Description	After a player joins to a game the role (Chaser or Mr. X) is determined.
Actors	New player
Includes	
Triggers	A player is joining a game.
Preconditions	
Postconditions	A role is assigned to a player.
Flow of events	
Alternative paths	

Use case name	Choose role
Exceptions	
Notes	If the role of Mr. X is unassigned at joining time of a new player it is possible to apply for the role of Mr. X. Possibility to adjust game rules for a game with more players (about 10 or more): On the one hand a game could be limited to a defined number of players. If more players are available another game session will be started. On the other hand it could be possible to have two or more Mr. X. Therefore the game is not over if one Mr. X has been captured. But the player who captured Mr. X receives credits for his success.
Author and date	Mathias Kellerer, 12/12/06
Use case history	● v 0.1, initial version

8. Use cases – Player related

8.1. Move around

Use case name	Move around
Description	Every player moves around Munich using the predefined MVV subnet giving a ticket/token out of a limited number.
Actors	Chaser, Mr. X
Includes	
Triggers	
Preconditions	Chaser or Mr. X is located on a station on the MVV subnet map where it is possible to change the mean of transport.
Postconditions	After one move the Chaser or Mr. X is located on another station next to the starting position.
Flow of events	A usual move could consist of following steps: 1. Choose route and give ticket/token 2. Start moving 3. Use an appropriate means of travel 4. Arrive at destination and end move
Alternative paths	
Exceptions	
Notes	It is possible to use one of the following means of transport: ● Buses ● Subway ● Suburban railway ● Walking
Author and date	Mathias Kellerer, 12/12/06
Use case history	● v 0.1, initial version

8.2. Communicate with other players

Use case name	Communicate with other players
Description	In order to coordinate the movements (see 8.1 Move around) of the chasers a mechanism for communication is needed. Mr. X could also use this service e. g. to give hints about his current position to the chasers.
Actors	Chaser (primary), Mr. X (secondary)
Includes	
Triggers	Player wants to send a notification to the other players.
Preconditions	
Postconditions	
Flow of events	

Use case name	Communicate with other players
Alternative paths	
Exceptions	No communication is available for a long period of time. Therefore the message is out of date.
Notes	For the communication different appendages are possible: ● Text messages ● VOIP (difficult realisation) ● Voice messages ● Video messages
Author and date	Mathias Kellerer, 12/12/06
Use case history	● v 0.1, initial version

8.3. Leave game

Use case name	Leave game
Description	Every player (Chaser or Mr. X) has the opportunity to leave the currently attending game.
Actors	Chaser, Mr. X
Includes	
Triggers	A player does not want to play Munich Hunt any longer.
Preconditions	
Postconditions	
Flow of events	1. Chaser or Mr. X informs server about his intention to leave the game 2. Servers acknowledges 3. Chaser or Mr. X leaves game 4. Server informs other players 5. Game is balanced (if necessary)
Alternative paths	
Exceptions	
Notes	It should be considered that a re-entry is possible. Also it is possible that a player plays the game no longer actively and therefore the player will be excluded from the game.
Author and date	Mathias Kellerer, 12/12/06
Use case history	● v 0.1, initial version

9. Use cases – Chaser related

9.1. Catch Mr. X

Use case name	Catch Mr. X
Description	Every Chaser tries to catch Mr. X by cruising around Munich (see 8.1 Move around) depending on the last known position of Mr. X. In order to catch Mr. X several rules how to catch Mr. X are possible. Whether a Chaser has caught Mr. X correctly is determined by the server during synchronisation phase (see 5.2 Synchronise with server) which will be triggered by catching Mr. X.
Actors	Chaser
Includes	Identify Mr. X
Triggers	
Preconditions	Mr. X has not been caught before.
Postconditions	The game is over (see 4.5 End game session).
Flow of events	
Alternative paths	
Exceptions	
Notes	Different catching rules: ● Mr. X is within a specific range ● Chaser has to identify Mr. X with a photo (see 9.2 Identify Mr. X) ● Chaser and Mr. X have to be in the same location (a station on the subnet plan) within a defined period of time ● ...
Author and date	Mathias Kellerer, 12/12/06
Use case history	● v 0.1, initial version

9.2. Identify Mr. X

Use case name	Identify Mr. X
Description	In order to catch the right Mr. X it could be necessary not only to be in the same place within a specific range but also to identify an unknown person as Mr. X.
Actors	Chaser
Includes	
Triggers	
Preconditions	Game rules require an identification of Mr. X.
Postconditions	Ascertained whether the person is Mr. X or not.
Flow of events	
Alternative paths	

Use case name	Identify Mr. X
Exceptions	
Notes	The event of identification could be done in different ways: ● Every chaser has a photo from Mr. X and has to compare manually. ● A photo from the person is send to the server and the server decides whether Mr. X is shown on the photo or not. ● Identify Mr. X via his bluetooth fingerprint. ● ...
Author and date	Mathias Kellerer, 12/12/06
Use case history	● v 0.1, initial version

10. Use cases – Spectator related

10.1. Spectate at a game

Use case name	Spectate at a game
Description	Every person who wants to watch at a game Munich Hunt can connect to the http server provided by the game server and gets an overview of a currently running game.
Actors	Spectator
Includes	
Triggers	
Preconditions	
Postconditions	
Flow of events	
Alternative paths	
Exceptions	Too many spectators.
Notes	It is possible that spectators reside in Munich, watch a Munich Hunt game with a mobile device and decide to take part in this game (see 7.1 Join game).
Author and date	Mathias Kellerer, 12/12/06
Use case history	• v 0.1, initial version

11. Annotations

Possible game play improvements:

- Possibility for Mr. X to give hints to the chasers (see 8.2 Communicate with other players)
- Chasers could explicitly query the current position of Mr. X (only one or two times during a game session)
  - Mr. X would be informed about localisation by a chaser
- Increasing vibration alert could signalise closeness of a chaser and vice versa
- The actors “Chaser” and “Mr. X” could be represented by teams which are made up of several persons instead of playing on its own.
- GPS and UMTS would be the preferred technique because of frequent synchronisations (e. g. for better realisation of use case 9.1 Catch Mr. X). Therefore it will lead to better game play.