Seamful Design for Location-Based Mobile Games

Gregor Broll[1] and Steve Benford[2]

[1] Institute for Informatics, Embedded Interaction Research Group,
Amalienstr. 17, 80333 München, Germany
gregor@hcilab.org
[2] Mixed Reality Laboratory, The University of Nottingham,
Nottingham, NG8 1BB, UK
sdb@cs.nott.ac.uk

Abstract. Seamful design is a new approach to reveal and exploit inevitable technical limitations in Ubiquitous Computing technology rather than hiding them. In this paper we want to introduce its general ideas and apply them to the design of location-aware games for mobile phones. We introduce our own seamful trading-game called "Tycoon" to explore seams on this platform and show how to incorporate them into the design of mobile games. We want to evaluate how applications for the mobile phone platform which use cell-positioning can exploit seams for better interaction, gameplay and usability.

1 Introduction

Devices and applications for mobile communication and computing usually suffer more from their technical limitations than other systems of Ubiquitous Computing. Patchy network coverage, fluctuating signal strength, deviations in positioning and the generally limited resources provided by mobile devices are an everyday reality for their users. Usually they experience these limitations indirectly as sketchy and slow mobile internet access, variations in the quality of speech transmission, loss of connections or ambiguities in positioning. As mobile applications and services will offer more sophisticated and demanding services e. g. for multimedia and entertainment, those seams in both technology and interaction-design are even more likely to come to the users' attention.

Despite the efforts of seamless design to smooth over these bumps and cracks in ubicomp systems and their infrastructures using costly investments in better and more reliable technology, we think that mobile applications could greatly benefit from exploiting those seams rather than abandoning them. Contrary to seamless design, seamful design tries to reveal inevitable seams in ubicomp systems and use them to increase the awareness for system infrastructures, their heterogeneous components and otherwise neglected yet useful information within the system.

We want to explore the possible benefits of seamful design and show how to apply it to mobile applications in general and mobile games in particular as they provide a very flexible and playful environment for an easy evaluation of seamfulness. We in-

F. Kishino et al. (Eds.): ICEC 2005, LNCS 3711, pp. 155 – 166, 2005.

troduce our own design for a seamful location-based game for mobile phones called "Tycoon" and want to explore how to exploit the seams on this platform – especially in cell-based positioning, network coverage and data inconsistencies – as resources for better usability, interaction and gameplay.

2 Designing Ubiquitous Computing Systems

2.1 From Seamlessness to Beautiful Seams

Both seamfulness and seamlessness can guide the design of ubicomp systems and are more or less derived from Mark Weiser's vision of Ubiquitous Computing. As part of this vision Weiser called for invisibility as a general design goal of Ubiquitous Computing and especially for invisible tools that don't intrude on the user's consciousness but let him focus on the task and not the tool itself (see [1]). According to [2] this postulation seems to have "been translated into requirements for seamless integration of computer system components, as well as the interactions supported by those components". Since then seamless design became the ruling paradigm for realizing Ubiquitous Computing systems.

The infrastructures of ubicomp systems and applications usually contain many different heterogeneous components and devices. Seamless design advocates knitting these components tightly together and assembling them into one single entity in order to hide the complexity of the infrastructure caused by the heterogeneity of its components. The invisibility of the individual parts enables seamless interaction and allows users to unconsciously interact with them through the whole entity which itself becomes "literally visible, effectively invisible" [1].

An everyday example of interacting with a (seemingly) seamless system is the handover between adjoining GSM-cells while using a mobile phone. Though being an essential feature of mobile communication technology, handover between cells is kept hidden in the infrastructure of service provider networks. Because of this, handover is automatically and seamlessly handled while people simply use their mobile phones unaware of handover, changing cells or even the concept of cells.

Despite its benefits of comfort and simplicity, the paradigm of seamlessness is questioned. [3] points out that Mark Weiser actually opposed it as a misleading concept. Instead of making everything the same, reducing different components in a seamless system to the level of a "lowest common denominator" [4] and sacrificing their uniqueness for the goal of overall compatibility, he calls for "seamful systems" which [2] paraphrases as: "making everything the same is easy; letting everything be itself, with other things, is hard". Weiser also advocates seamful systems with "beautiful seams" [4] meaning that seamfully integrated parts of a system could still provide seamless interaction with the whole system while openly retaining their individual features.

2.2 Seams and Seamfulness

An approach to seamfulness and seamful design becomes even more intelligible as infrastructures of ubicomp systems still have limitations and boundaries despite their efforts for seamless integration and interaction.

Just as seamless design refers to both the technical joints between infrastructure-components and the overall experience of smooth, seamless interaction, the concept of seams in ubicomp systems comprises the technical cracks and bumps in the joints between those components as well as the user's experience of them in the system. Seams can be seen as deviations in actual use from a notional ideal of technological continuity or uniformity including discontinuities in technologies themselves and discontinuity between what actually happens and what the system observes.

Seams in ubicomp systems are mostly caused by technical limitations and differences between heterogeneous components in the underlying infrastructures. Among them are finite resolution and sampling rates of sensors, fluctuating signal strength, limited connectivity, defective transformations or delays in communication. These technical constraints in the infrastructure come to the user's attention when they interact with a supposedly seamless system. Seams then reveal themselves as uncertainties, ambiguities or inconsistencies.

Probably the most common seams can be found in systems for navigation and mobile communication where they include inaccurate positioning, sketchy internet-access or patchy network coverage. For example infrastructures for outdoor- and indoor-positioning with GPS- or ultrasonic-trackers are notorious for their literally built-in deviations. These are due to technical constraints like finite coarseness of sensing-technology as well as non-technical restraints like "urban-canyons" or reflecting surfaces which make accurate sensing difficult. People using positioning systems experience those seams as uncertainties about their current position.

2.3 Seamful Design and Designing for Appropriation

Often enough seams can't be completely hidden and undermine the ideal of a totally seamless system. Instead of trying to iron these bumps out of an ubicomp application, seamful design tries to incorporate its heterogeneous components while recognizing and maintaining their characteristics and uniqueness without giving up the overall goal of seamless interaction. In order to accomplish the goal of "seamless interaction but seamful technology" [5] outlines the process of seamful design by identifying its 3 key-problems as "understanding which seams are important", "presenting seams to users" and "designing interactions with seams".

The first step to turn seams into a helpful feature is to identify those seams that can enhance a system's functionality, understand them and find out how they can become a resource for user-interaction. Seamful design reveals and recognises the infrastructure's influence on user-interaction and increases the awareness for features that cause seams in applications. If designers know how certain seams affect interactions, they can decide how to incorporate them into an application, how to channel their effects into useful features and include them in the process of interaction design itself. That way seamful design allows users to use seams, accommodate to them and even exploit them for their own advantage.

The less the presentation of seams is restricted in the way people can interact with them, the more can users harness those seams and adapt to them. Giving them the opportunity and freedom to play with seams, explore how to use and exploit them in new ways adds a flexibility to seamful interaction that increases the depth of an application. This new quality may ultimately lead to the more general concept of

designing for appropriation. Users are given a presentation of seams that increases their awareness for them but are not restricted to interact with them in a compulsory way. Design for appropriation allows users to interact with seams individually, take advantage of the gaps and limitations in ubicomp infrastructures and develop new patterns of behaviour that have not been considered during the initial design of the system.

The use of mobile phones offers intuitive examples of appropriation: Among their most common seams are patchy network coverage and local variations in signal strength which are commonly accepted as reasons for not answering or dismissing a call. Users can exploit their knowledge about those seams and adapt their behaviour by pretending to be outside of network coverage (e. g. in tunnels) or to have a bad reception which - unfortunately - forces them to dismiss a call.

2.4 Bill – A Seamful Mobile Game

Being one of the first seamfully designed mobile games, "Bill" was developed at the University of Glasgow by Matthew Chalmers et al. [6], exploring seams in 802.11 wireless networking. Bill is a seamful mobile multiplayer game in which two teams of players compete against each other by collecting virtual coins in a designated gaming-area. This area features wireless access points connected to the game server which distributes information about the game. Each player is equipped with a PDA supporting GPS-positioning and 802.11 wireless internet-access. The GUI of Bill displays information about the locations of coins, players and network covered areas on a pan- and zoomable map of the gaming-area (see Fig. 1).

During the game, players roam the gaming-area, discover access-points and collaboratively build up a shared map of network covered areas. Local coverage and quality of wireless network access are displayed as transparent squares on the GUI's map with different colours indicating different levels of signal strength. Players use GPS-positioning to collect coins but have to get inside the coverage of a wireless access point in order to upload them to the game server and earn credits for their team. A simultaneous, collaborative upload with several other team-mates earns players a cumulative amount of credits for the same amount of coins. An additional pick pocket-feature lets players steal coins from each other if thief and victim are close enough (according to GPS) and within access point coverage.

The main seams in Bill are the patchy coverage of the wireless access points and the dynamic boundaries of the network covered areas. Presenting information about them to players as a part of the game's interface turns the negative seams into a helpful visual feature of the game's interface. Players can use it and develop an understanding of network coverage and signal strength in order to succeed in the game. They can even use this information to adapt their behaviour by staying out of network covered areas in order to avoid pickpockets, by knowing where to gather with team mates for a collective upload or by specializing as pickpockets themselves.

Other seams in Bill include inaccurate GPS-positioning that allows players to collect coins which are actually out of their reach and data inconsistencies that let different players find the same coin independently from each other as they see different subsets of the available coins at different times unless they are regularly updated.

Fig. 1. Bill's GUI includes general information about the game (e.g. scores), icons for invoking interactions (e. g. "Upload", "Pickup") and a map of the gaming-area with the positions of coins as well as colored squares visualizing wireless network coverage and signal strength (screenshots taken from [6], [7])

3 Seamful Design for Mobile Phone Games

3.1 Introducing Tycoon

The "Bill"-game showed how to turn the seams in a mobile application supporting GPS and 802.11 into helpful features of a game, how seamful design can drive its gameplay and how players can exploit seams in order to win. We want to extend the ideas of seamful design to games for mobile phones, explore which seams occur on this platform and how to apply seamful design in order to exploit them for better gameplay, interaction and appropriation.

For this purpose we developed our own seamful game for mobile phones called "Tycoon". Tycoon is a location-based multiplayer trading game with a simple producer-consumer-cycle that uses the different GSM-cells of a service provider network within a designated gaming-area, e. g. the centre of a city. Each of these cells in the physical area is virtually mapped to either a producer or a consumer in the game (see Fig. 2). Tycoon uses the metaphor of a wild west scenario to communicate its central mechanisms of collecting resources from producers which are called "mines" and using them to buy objects from consumers which are called "brokers" and have the names of cities or counties in California (see Fig. 2). These cells are called "brokers" because they sell objects in their areas in exchange for collected resources.

While playing the game, players are travelling between the cells in the gaming-area, collect local resources in mines, use them to buy global objects from brokers and get credits for claiming them. Each mine produces an unlimited amount of one of the

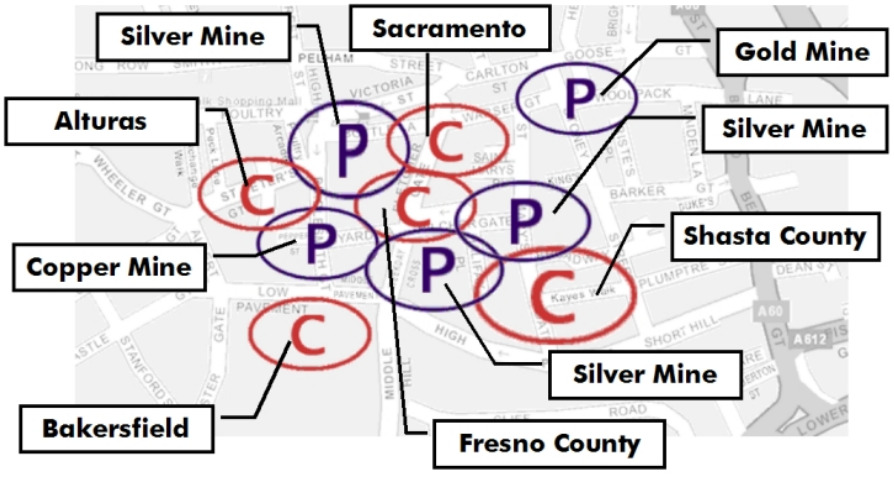

Fig. 2. Schematic example of the GSM-cells within a gaming-area and their mapping to mines (producers) and brokers (consumers) of the game

three *local resources* in the game – gold, silver or copper. They are called local resources because players can collect them independently of each other and don't compete for them. Each broker has a list of *global objects* e. g. different buildings or estates in towns and counties that players can buy with their local resources. There is only a limited number of unique global objects in the game (e. g. there is only one saloon in Sacramento) and players compete against each other for claiming them, since every global object can only be claimed once by one player. Each global object has a combined price of two local resources and a value in credits. Players have to enter a broker's cell and pay an object's price in order to claim it and earn its value in credits. The objective of the game is to gather as many credits as possible; it ends when all objects are sold and the player with the most credits wins.

The responsibility for the maintenance of the game-state and its overall correctness is split between the global game-server and the local clients on players' mobile phones. These clients tell players where they are, help them navigate through the gaming-area and supervise the collecting of local resources. The clients connect with the game-server via GPRS in order to buy global objects from single brokers and ask for updates on the *global game-state*, which comprises the general availability of all global objects of all brokers in the game. The game-server processes and answers client-requests, manages the global game-state and runs the game by controlling its logic. It doesn't know anything about individual resource-collecting of different clients and clients only have a limited knowledge about the global game-state unless they ask the game-server. In short, the cell-phone clients each manage individual local game-states and the game-server manages the shared global game-state.

3.2 How to Play Tycoon

Unlike Bill Tycoon doesn't provide a general map of the gaming-area including the locations of the different mines and brokers. Mobile phone displays are often too

small to be useful for map-navigation and a map of the Tycoon gaming-area would be significantly bigger than the map in Bill. We want the players to start Tycoon by having to explore the gaming-area and discover mines, brokers and their locations by themselves. That way the players can gather their own knowledge about the gaming-area, where to find resources and where to claim objects.

Two features of Tycoon help players to navigate in the gaming-area: Whenever a player changes from one cell into another, an alert is triggered and he gets a notification about him entering a new cell. Afterwards the main screen displays the name of the new cell (Fig. 3).

Fig. 3. The main screen of Tycoon's GUI which is updated whenever a player crosses boundaries between adjoining cells. It is the central screen of Tycoon and shows the amount of collected resources and earned credits as well as the current location.

A log containing annotations about mines, brokers and their different locations is intended as an aid to memory in order to remember where their cells can be found. When a new log-entry is created, Tycoon automatically logs the current cell and the player can add a short description about the location (e. g. [gold mine] near book shop). During the game each player will eventually build up his own log of locations.

Collecting local resources is only possible in suitable mines and the return depends on the time a player spends in the cell which is virtually mapped to a mine. A player can start and stop a counter on his mobile client and will be collecting the current resource until he decides to stop or until he leaves the cell. Collecting local resources is completely managed by the mobile client without the game-server's knowledge.

Each player starts Tycoon with a list containing the names of all brokers in the gaming-area but without showing which objects each of them is offering. A broker's initial list of objects will be revealed to the player once he discovered the appropriate cell and enters it for the first time.

In order to earn credits a player has to enter the cell of a broker, choose one or several entries from his list of available objects and claim them from the game-server. If his choice is still available according to the global game-state the player gets the object's credits added to his account. If objects have already been claimed by other players and are no longer available the player keeps his resources for future claims. In either case the player gets an update on the availability of all objects for the current broker.

Players can ask the game-server for a separate update on the global game-state anytime and anywhere in order to soften the possible inconsistencies between the

game-server and the mobile clients concerning the availability of global objects. An update has to be paid with a certain charge in local resources since it gives players a considerable advantage in the game. A player will receive the update only for the lists of global objects of brokers whose cells he has already discovered. This way no player can skip the discovery of all the brokers by simply getting an update.

While a player can ask the game-server for updates anytime and anywhere, collecting resources and claiming objects are location-dependent functionalities of Tycoon and are only available when a player is within an appropriate cell.

3.3 Understanding Seams in Mobile Phone Applications

The design of Tycoon tries to cope with three seams which we consider to be characteristic for the mobile phone platform especially when compared to a seamless approach: dynamic cell-coverage, expensive internet-access and data-inconsistencies.

Usually mobile phone users are unaware of their current cell when using their phones, since the handover between different cells is handled seamlessly. Contrary to the symbolic presentation in Fig. 2 the coverage of GSM-cells is not static and has no neatly defined borders. Fig. 4 shows coverage and propagation of GSM-cells in an area of London based on samples of cell-ids and their GPS-positions. Their coverage is depending on many factors, cells' boundaries and propagation are rather dynamic and fluctuating and as Fig. 4 shows, cell-coverage has irregular shapes and adjoining cells often overlap and don't share exact borders.

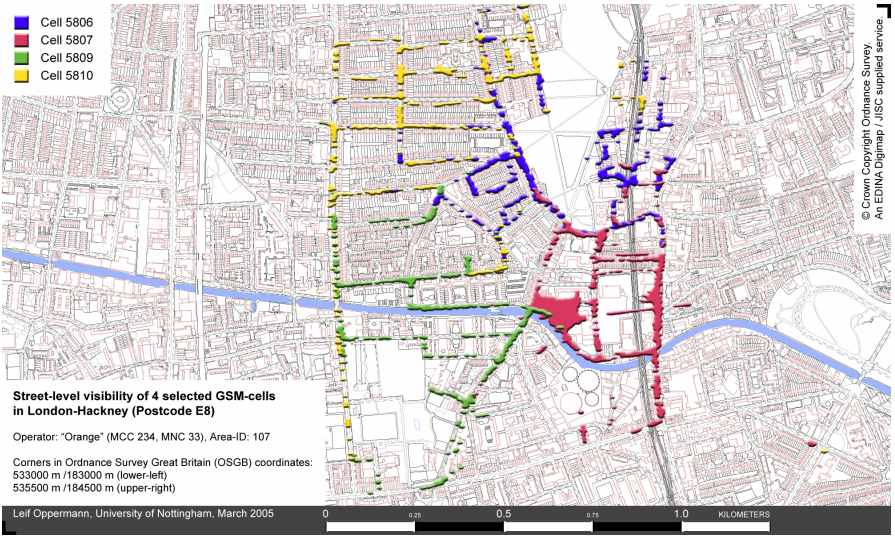

Fig. 4. Samples of cell-ids and their GPS-positions in London

The invisible handover between cells is handled seamlessly so that users don't get any information about their current cell and don't have to worry about their position, dynamic cell coverage and propagation or flipping cells. While seamless design usually hides this information, we would like to present it to players and make them

aware of this information so that they can take advantage of it during the game and use it as a valuable resource. In a location-based mobile game players are dependent on knowing where they are and dynamic boundaries and propagation of cells may raise interesting design-issues concerning the influence of positions and relations between GSM-cells on the behaviour of the users during the game.

Tycoon is a multiplayer game for cell phones and the driving part of its gameplay is the competition between several players with mobile clients. In order to compete with each other including asking for claims, having the availability of objects confirmed, scores compared or the end of a game announced, players have to be able to turn to a central entity which controls a shared global game-state. Mobile internet-access is most convenient and almost indispensable for accessing a shared game-server on the internet and synchronising its data with the mobile clients.

A seamless approach to this synchronisation would try to guarantee as little inconsistencies between the global and all local game-states as possible. That would mean that whenever a client changes the global game-state e. g. by claiming an object, the server would have to route this change to all other clients or they would periodically have to ask the game-server for an update on the latest changes.

Either way a lot of traffic would be generated and mobile internet-access via GPRS is still quite expensive. Despite the above definition of seams, these expenses are neither rooted in technical limitations nor show themselves as uncertainties or ambiguities and may therefore be considered as an artificial seam. But it still constrains the uniformity and continuity of a mobile application and we think that's why costly internet-connections can be treated as a considerable seam. We would like to turn this seam into a rewarding feature of the game by encouraging players to use less online-connections to the game-server without examining the technical limitations of GPRS-connections like delays or its general availability.

The question of whether to use more or less GPRS-connections to synchronise game-states goes hand in hand with the probability of data-inconsistencies. Those would be an immediate consequence of insufficient synchronisation between a server and its clients. Inconsistencies could occur when individual clients synchronise data with the central server that maintains the global game-state which is shared between all clients. When one of these individual clients updates globally shared data, local copies of that data on other clients become inconsistent with the shared data.

As mentioned, a seamless approach to Tycoon would try to prevent data-inconsistencies through frequent updates. But when tackling the seam of costly GPRS-connections by making less of them, we also have to cope with the seam of increasing data-inconsistencies between mobile clients and the global game-server. While a seamless approach would try to guarantee a persistent globally correct game-state we want to enrich Tycoon's gameplay by turning inconsistencies into a gambling-element that rewards players for taking the risk of them during the game.

3.4 Combining Tycoon's Gameplay and Seamful Design for Appropriation

Tycoon uses the unique cell-id of GSM-cells in its gaming-area to support navigation and to determine its location-dependent behaviour during the game. The alert-mechanism is a monitor for the current location and its changes, it improves the visualisation of cells' boundaries and decreases the players' uncertainties about their

whereabouts by presenting otherwise neglected seamful information about cell coverage and propagation. It is an effective and flexible means of navigation in the gaming-area especially since the propagation of its virtual areas dynamically changes with the coverage of the physical cells they are mapped to. A static map of the gaming-area would need an expensive mechanism to continuously sample cell coverage and display their propagation in real time.

The alert-mechanism makes players not only aware of dynamic changes in the cell-propagation but is also part of Tycoon's interaction with the seam of dynamic cell-coverage. Tycoon's behaviour is partly location-dependent. It changes with the current location and adapts to it, e. g. collecting local resources automatically stops when a players leaves a mine and enters another cell. The functionalities of collecting resources and claiming objects are exclusively available in mines or broker-cells. The log that helps players remember where to find different mines and brokers only logs information in cells within the gaming-area.

Like in the Bill-game players can use their spatial knowledge about the gaming-area to adopt their own strategies of how to move between cells and how to find the most efficient tactics of which resources are needed to buy which available objects and where to find them in nearby mines. They can also exploit ambiguities caused by dynamic cell propagation and boundaries more effectively when they find an area where adjoining cells overlap and flip after some time without moving. This is also an interesting issue for designing the gaming-area of Tycoon and assigning mines and brokers to different cells.

In order to cope with the seams of expensive internet-connections and data-inconsistencies, Tycoon encourages players to spend more time offline and synchronize their local game-state (individual clients' knowledge about available global objects) less often with the game-server's global game-state (the overall availability of global objects). Its trading-mechanism features a simple economic model that offers players an incentive to engage in extended offline-play and spend more time on collecting a greater number of (more valuable) resources.

As mentioned before, global objects each have a price in local resources and earn a player a certain amount of credits when claiming them. Both values are related to each other as the credit-values of objects rise with their prices in local resources. The local resources have different values themselves which is indirectly expressed by the time it takes to collect one unit of each (2,4,8 seconds to collect 1 copper-, silver-, gold-nugget). That way the credit-value of a global object is also related to the time it takes to collect the resources that are necessary to buy the object. But the values of resources, the values of objects and their prices don't rise proportionally. The economic model is tuned in a way so that the more time a player spends offline to collect more and more valuable resources in order to afford more and more valuable objects the more credits per second could he possibly earn than by collecting less resources for smaller, less valuable objects during the same time. Additionally a player gets a discount for successfully claiming several objects from a broker with the same request to the game-server. Of course in order to afford buying several objects, players have to spend even more time offline to collect more and more valuable resources.

The goal of this economic model is to give players an incentive to spend more time offline between connections to the game-server and collect local resources which is

more profitable than spending less time offline collecting less and less valuable resources for claiming less valuable objects.

Extended offline-play increases the danger of inconsistencies between the global game-state and a player's local list of available global objects. Other players can claim objects on that list which are then no longer globally available but are still shown as available on a player's local list. Since players can't rely on getting constant updates of the global game-state which would require regular expensive connections to the game-server, they have to consider the growing probability of inconsistencies between local and global game-state when deciding how much time they spend offline for collecting local resources.

This is where the gambling-approach of Tycoon comes into play: It recognizes these inconsistencies in the gameplay and rewards players not only for spending more time offline but also for taking the risk of inconsistencies. Players are free to collect as many local resources as they want but can only turn them into credits when they successfully buy and claim global objects with them. The more time they spend offline, the more profit is possible but the greater becomes the probability of inconsistencies. Players have to adopt their own individual strategies and consider their chances of earning more credits during the same time against the risk of data inconsistencies.

4 Conclusion

Seamful design is a rather new approach to ubicomp applications other than seamless design. We showed how to use and exploit seams on the mobile phone platform by incorporating them into our own seamful game. Instead of hiding these seams we revealed them and integrated them into the game's interaction design for a better gameplay. Based on our experience with designing and implementing a seamful trading-game we think that seamful design is a rewarding approach for realizing applications of Ubiquitous Computing and a considerable alternative to seamless design whose efforts often result in expensive improvements of infrastructure-technology but not always in better applications.

Acknowledgements

This project was supported by EQUATOR, an Interdisciplinary Research Collaboration (IRC), funded by the UK Engineering and Physical Sciences Research Council.

We would like to thank Leif Oppermann (MRL, University of Nottingham) for providing us with a map of cell coverage and propagation based on GSM-samples and their GPS-positions taken during a workshop in London. We would further like to thank Albrecht Schmidt (Institute for Informatics, Embedded Interaction Research Group, Munich) and Holger Schnädelbach (MRL, University of Nottingham) for their support and making this project possible.

References

1. Weiser, M.: *The world is not a desktop*. ACM Interactions 1(1) (1994) 7-8.
2. Chalmers, M., MacColl, I.: *Seamful and Seamless Design in Ubiquitous Computing*. Technical Report Equator-03-005, Equator [Technical Reports] (2003)
3. Chalmers, M., MacColl, I., Bell, M.: *Seamful Design: Showing the Seams in Wearable Computing*. Eurowearable 2003, University of Birmingham, UK. International Conference Proceedings (2003)
4. Weiser, M.: *Creating the invisible interface (invited talk)*. ACM Conf on User Interface Software and Technology (UIST94) (1994), 1
5. Oulasvirta, A.: *Notes on Seams, Seamfulness and Seamlessness*. http://www.hiit.fi/u/oulasvir/Haninge (2004)
6. Chalmers, M., Bell, M., Hall, M., Sherwood, S., Tennent, P.: *Seamful Games*. http://www.ubicomp.org/ubicomp2004/adjunct/demos/chalmers.pdf (2004)
7. Chalmers, M., Bell, M., Brown, B., Hall, M., Sherwood, S., Tennent, P.: *Seamful Game*. http://www.seamful.com (2004)