Interaction Design for a Mobile Context-Aware System Using Discrete Event Modelling

Annika Hinze **Petra Malik** **Robi Malik**

Department of Computer Science, University of Waikato, Hamilton, New Zealand
E-mail: {hinze,petra,robi}@cs.waikato.ac.nz

Abstract

This paper describes our experience when applying formal methods in the design of the tourist information system TIP, which presents context-sensitive information to mobile users with small screen devices. The dynamics of this system are very complex and pose several challenges, firstly because of the sophisticated interaction of several applications on a small screen device and the user, and secondly because of the need for communication with highly asynchronous event-based information systems.

In a first step, UML sequence diagrams have been used to capture the requirements and possible interactions of the system. In a second step, a formal model has been created using discrete event systems, in order to thoroughly understand and analyse the dynamics of the system. By verifying general properties of the formal model, several conceptual difficulties have been revealed in very early stages of the design process, considerably speeding up the development. This work shows the limitations of typical methods for interaction design when applied to mobile systems using small screen devices, and proposes an alternative approach using discrete event systems.

1 Introduction

Interaction design for mobile location-aware systems is a very challenging task that can only to a limited extent draw from experiences in designing desktop-based systems (Kjeldskov, Graham, Pedell, Vetere, Howard, Balbo & Davies 2005). We are implementing TIP, a mobile tourist information system that provides context-aware information delivery to travellers. Currently, the software is being redesigned into a service-oriented architecture to support the system's various services in a flexible way.

The complex interactions between the TIP core system and its various services together with the management of user context (such as location and interests), and the interplay with the mobile device create several design challenges (Hinze & Buchanan 2005). Typically, the design of mobile systems focuses on interface issues, while the design of location-aware systems, such as tourist guides, often follows a user-centred design combined with a-posteriori usability studies.

In contrast, we follow a formal design approach to address the service interaction issues. We use automata for discrete event modelling (Cassandras & Lafortune 1999, Ramadge & Wonham 1989) since this design methodology mirrors the abstraction concept that guides the system's internal control using event-based interaction.

Traditionally, formal methods have mostly been applied to mission-critical software. This paper shows the advantages of using a formal approach in the design of a mobile location-aware system. We provide a complex formal model of the TIP system and show how its analysis leads to a much improved system architecture.

Section 2 provides a brief introduction to the TIP system and its architecture. Section 3 illustrates the design of the interactions within and between the services using UML sequence diagrams. Section 4 shows the interaction design by modelling interactions with automata for discrete event systems. Section 5 briefly refers to related work. In Sections 6 and 7, we conclude the paper with a discussion of the lessons learned regarding interaction design for mobile systems, and a summary of our findings and future work.

2 The TIP System

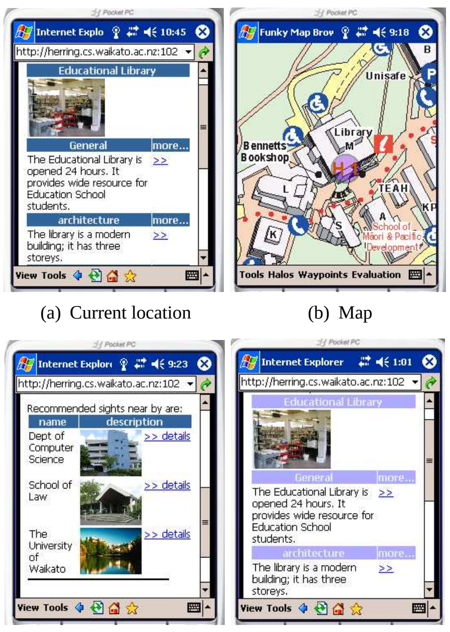
TIP is a mobile system that delivers tourist information about sights based on the user's context: their location, two personal profiles describing interests, and the user's travel history. The system also considers the sights' context, such as their location and memberships in semantic groups. Sights that are in a semantic group share certain features, e.g., medieval churches. Details about the TIP core can be found in (Hinze & Voisard 2003). In addition, TIP supports several services such as recommendations and a map service (Hinze & Junmanee 2005, Hinze & Buchanan 2006).

This section briefly illustrates TIP in use, argues for a new service-oriented architecture for TIP 3.0, and describes the basic interactions within the architecture.

2.1 TIP in Use

Figure 1 shows the TIP core system and its services in use. In Figure 1(a), sight-related information is delivered for the current location of the user. TIP also supports navigation by maps that dynamically update the current location and the location of nearby sights. Figure 1(b) shows the map for the same location as before. For recommendations, we assume that a user who has seen several sights in a group is interested in seeing more (see Figure 1(c)). Recommendations are also given based on user feedback and profiles. Users may also browse for places they are not currently visiting—this is indicated by a different colour scheme as shown in Figure 1(d).

The TIP system combines an event-based infrastructure with a location-based service. The heart of the system is the filter engine cooperating with the location engine. The filter engine selects appropriate information from different source databases depending on user and sight context. Thus, the system's interaction logic is defined in context-aware profiles for the event-based filter engine. The system is implemented as a client/server architecture, supporting desktop clients as well as mobile clients on a hand-held device with appropriate interfaces.


Internet Explo 10:45
http://herring.cs.waikato.ac.nz:102
Educational Library
General more...
The Educational Library is opened 24 hours. It provides wide resource for Education School students. >>
architecture more...
The library is a modern building; it has three storeys. >>
View Tools

Funky Map Brov 9:18
Unisafe
Bennetts Bookshop
Library
TEAH
School of Māori & Pacific Development
Tools Halos Waypoints Evaluation

Internet Explor 9:23
http://herring.cs.waikato.ac.nz:102
Recommended sights near by are:

name	description
Dept of Computer Science	>> details
School of Law	>> details
The University of Waikato	>> details

View Tools

Internet Explorer 1:01
http://herring.cs.waikato.ac.nz:102
Educational Library
General more...
The Educational Library is opened 24 hours. It provides wide resource for Education School students. >>
architecture more...
The library is a modern building; it has three storeys. >>
View Tools

(a) Current location (b) Map

(c) Recommendations (d) Browse location

Figure 1: TIP in Use: Context-dependent information delivery; core system information delivery (a), services (b and c), and browsing interface (d)

2.2 Towards a Service-Oriented Architecture

The current version 2.9 of TIP is a client/server system that is being extended to a hybrid client/server and peer-to-peer structure in TIP 3.0.

The apparently simple final service provided to the user by TIP 2.9 (a tourist guide with map, sight data, and recommendations) is in fact a composition of a variety of services (Hinze & Buchanan 2006). These services need to communicate with each other both within the same machine and between computers, using thick- and thin-client scenarios, and occasionally hybrid approaches. Existing mobile information systems often follow a monolithic approach. Our modular, service-oriented approach allows for the exploration of alternatives, e.g., of communication or implementation. Services can be provided in different ways or add new features without requiring changes to existing services.

With TIP 2.9, we have created a framework in which mobile services cooperate (e.g., the map and the information display system to provide sight indicators on the map). For TIP 3.0, further forms of cooperation and composition are needed. For example, we plan to extend the map to support different halo cues (e.g., for sight type or recommendation), which requires advanced inter-process communications.

With the trend of information systems increasingly being accessed using mobile devices and small screen interfaces, the aforementioned requirements will become more pronounced. Client devices will provide a number of pre-installed services and users will add their own selections. Consequently, we believe that for TIP 3.0 even stronger decoupling and modularisation is needed: a mobile infrastructure for mobile information services needs to flexibly support existing, changing, or new services. The design of TIP 3.0, for which the interaction design is reported in this paper, will see the redesigning TIP into a Service-Oriented Architecture (SOA) using web services.

2.3 Basic Interactions

Figure 2 illustrates the distributed nature of TIP's services and components. The databases and basic services are

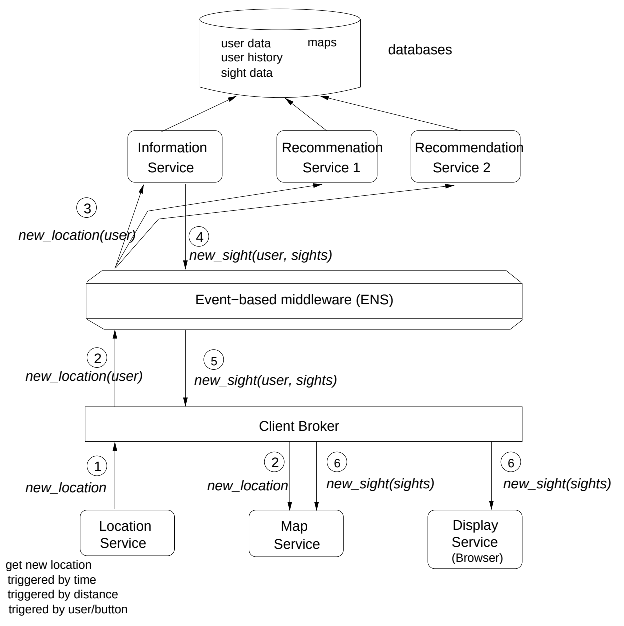


Figure 2: TIP 3.0 architecture and interactions between services and components

held on the server side (Figure 2, top). Each mobile client can have several services running. Interaction between services on the mobile device is coordinated by the client broker (Figure 2, bottom). The middle layer is formed by an event-based communication middleware. We abstract from details of the implementation on server side.

To clarify the architecture, we now explain the basic interaction sequence of the TIP system after a user's location change. This sequence is indicated using circled numbers ① to ⑥ in Figure 2; it is also specified by the sequence diagram in Figure 3.

First, the *location service* sends the current location of *this* user. This may be triggered by time (e.g., send every minute) or by location (e.g., send if the user moved more than 500m) or by user interaction with the system interface (e.g., a button being pressed to retrieve the new location). In Step ①, this information is sent to the client broker that coordinates all services on this mobile device.

In Step ②, the *client broker* attaches the user ID and sends the information to the event-based middleware (ENS). The information is also sent to the map service that is running locally on the mobile device. Here, we abstract from the possible distribution of the ENS over the mobile clients and the server. The *event-based middleware* holds service registrations and profiles about the information need of each service. As a basic service offer, it sends the information about the user's current location to the *information service* (and to all other services that requested this information). This is shown as Step ③. The event-profile logic that can be used for the ENS in this context is introduced in (Hinze & Voisard 2003). The information service receives the user's location (`new_location`) and the user ID; it looks up the sights nearest to the location that are of interest to the user.

As Step ④, the information service sends back to the ENS the list of sights that are close to the user (`new_sight`). The ENS forwards that information to the user's mobile device (i.e., to the client broker); shown as Step ⑤. The client broker distributes the information to the local services as needed, in Step ⑥. In this case, the *display service* receives the list of sights and sight information; it will display this information to the user to choose the sight in which they are interested.

The *recommendation service* also receives the user's location (`new_location`) and the user ID in Step ③. It looks up the sights that would be of interest to the user and

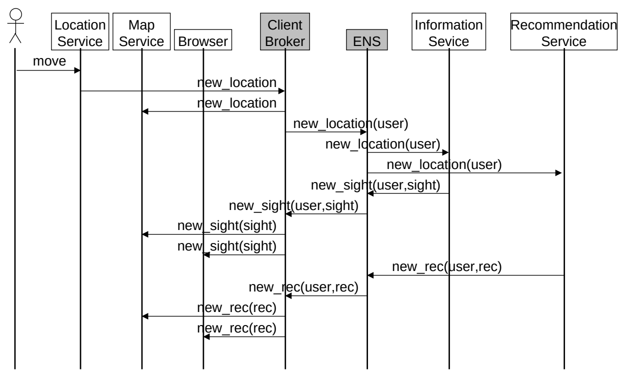


Figure 3: Basic interaction: User change of location. As the user moves, browser and map are updated with location, sights and recommended sights (shaded services are located on the middleware layer)

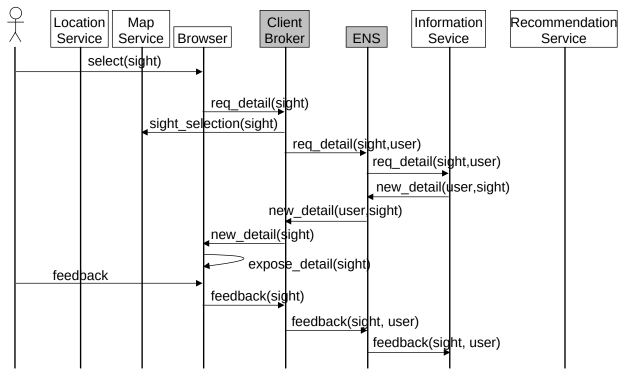


Figure 4: Select sight from the list for more information. Interaction follows after interaction in Figure 3

which the user has not seen already. Recommendations may also use feedback from other users. The functionality of the recommendation services is described in detail in (Hinze & Junmanee 2005). As shown in Figure 3, the recommendation service sends a list of sights back to the ENS. This list is then forwarded to the client broker on the mobile device. The client broker distributes the information to all services that are interested, i.e., the display service (for profiling information about the sights) and the map service (for highlighting the recommended sights).

In addition to this basic interaction, the TIP system needs to support several other interaction sequences. These are described in the next section.

3 Interaction Design Using Sequence Diagrams

This section describes several scenarios of possible interactions of the users and services in the TIP system. UML sequence diagrams (Kent 2001, Object Management Group 2003) are used, because they are a commonly understood means to describe interaction sequences. We focus on the interactions of a mobile client with the server, and abstract from the possible distribution of the data across other peers.

The scenarios presented in the following are grouped into basic interactions that describe the standard functionality of the TIP system, refined interactions that elaborate the details of how location information is to interpreted, and interface interactions that explore the specifics of user interaction on a mobile device.

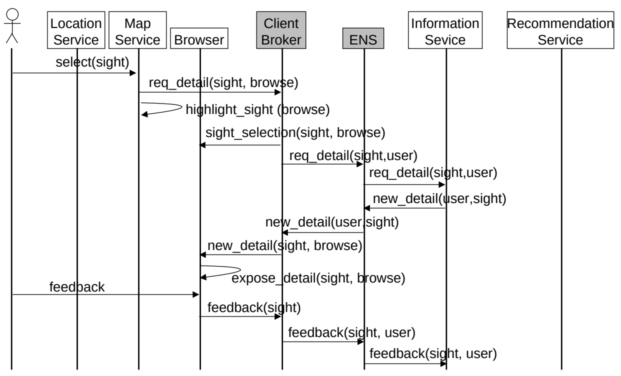


Figure 5: Select (distant) sight information, e.g., about recommended sights, from the map

3.1 Basic Interactions

1. **User change of location (see Figure 3):** The user changes their location and, subsequently, browser and map are updated with location, sights and recommended sights. This interaction is described in detail in the previous section.

2. **Select sight for more information (see Figure 4):** The user picks the sight they are looking at from the list of sights presented to them on the screen or from the map. If necessary, the application switches over to a browser interface when the information is available. Users can give feedback about how they liked the sight. Moving the mouse over the sight in the map without clicking could give some information (already downloaded in the task in Figure 3).

3. **Select sight information for distant sights / browse (see Figure 5):** An example is the browsing for more information about recommended sights (from the browser or the map). The design in Figure 5 (selecting on the map) has the drawback that people can assign feedback to sights they did not visit. Another option would be to present a list with all sights visited and not rated, and another with all sights visited to redo the ratings. The given design is useful to set up the system when used the first time—users can rate sights they know, even though they did not have the system yet.

4. **Browse for new locations on the map:** The user can also explore new areas of the map, placing a virtual user onto a new location, to receive information about that place without having to be at that location. Zooming and moving of the map is equivalent to scrolling in a browser; all interaction stays within the service. Location-browsing, as modelled here, is the equivalent of browsing on information pages as modelled in Task 3.

3.2 Refined Interactions

The modelling of location transmission can be refined to capture different approaches. This opens new possibilities for interaction design.

5. **New location transmission (see Figure 6):** So far, we abstracted from the different modes of sending locations. As indicated in Figure 2, location is measured and transmitted constantly (we ignore here the internal control of the location signal, such as GPS); the transmission of the location information to other services depends on the mode: location can be sent (a) continuously whenever the user moves, (b) depending on time since the last reading, (c) depending on distance from last reading, and (d) triggered

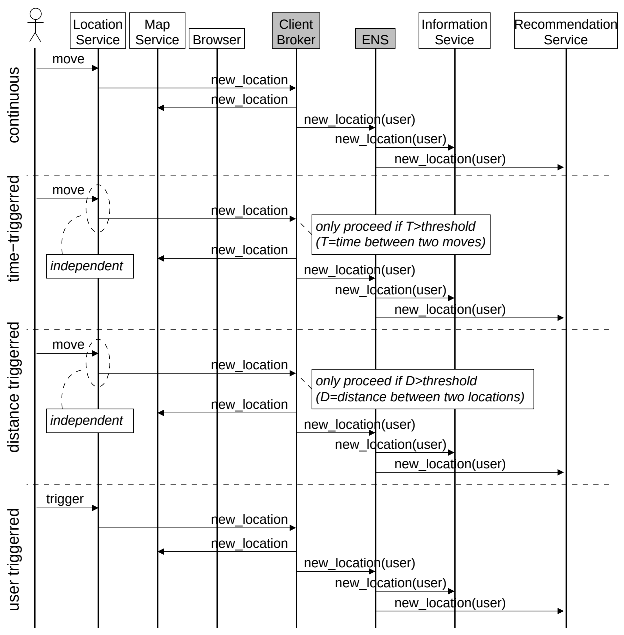


Figure 6: Different variations for location transmission

by user action (e.g., button pressed). Hybrid forms are also conceivable and depend on the actual location service implementation (e.g., location signal is sent according to time frequency or if the distance crosses a threshold). In fact, the system may have to behave differently for different services involved (see Task 6). Currently, TIP 2.9 supports the latter mode of requiring user interaction with the system, which leads to several irritating features for the user interaction.

Modelling the different modes using sequence diagrams does not reveal their severe consequences for user interactions (such as a divergence between screen location and real location of the user without indication in the interface).

When the user moves, their representation (e.g., avatar or location-halo) in the map should move as well. This does not hold for the information delivered: We assume that new information is only shown when triggered by the user to avoid flickering of the display. Also, when information about a building is received, the user should be able to access this information while walking around the building.

6. **Browsing by walking (see Figure 7)**: This action depends on the implementation of Task 5 (transmission of location data). Switching into a different mode, users are able to browse through their environment by walking. On their screen, new locations and sights are continuously updated. Consequently, the interaction mode has to change for both interfaces (map and browser) when switching into this particular behaviour. One possible sequence diagram is shown in Figure 7: location information is continuously sent; the interfaces are updated whenever new information is available; the information system display is only updated on request. This mode is not designed for reading and browsing through content but for scanning the environment and discovery of new places.

3.3 Interface Interactions

The interactions for navigating on the screen and in the information history cannot be modelled satisfactorily using

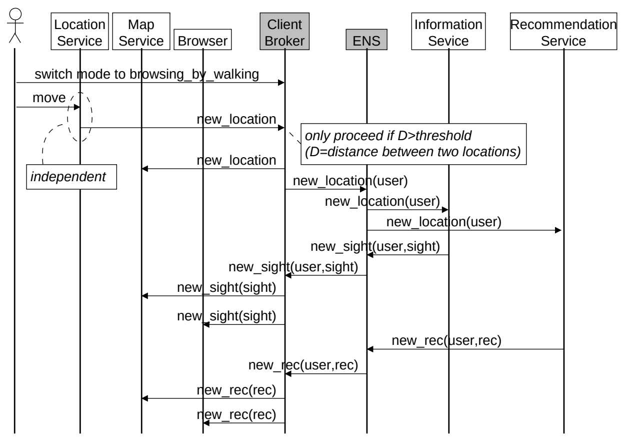


Figure 7: Browsing by walking

sequence diagrams. The different states of the interface are an important aspect that is missing in these models.

7. **Interaction: To current location (Synchronise)**: On pressing this button, the user interface returns to the current location of the user on the map. The location is newly measured and sent to the system; that is the user and the system are synchronised. This button is helpful if the user was browsing through information or zooming and navigating on the map.

8. **Interaction: To last location ('Back')**: On pressing this button, the user interface returns to the last location that was measured, ignoring user moves in between. This gives the user the opportunity to return to the last information that was delivered before they started browsing and/or walking around. It also offers the option of moving backwards through locations instead of presented pages.

9. **Interaction: Navigation 'Back'/'Forward'**: On pressing this button, the user can browse backwards through the pages that have previously been presented to them, or move forward from earlier pages. This navigation should work for all services (each service itself may provide their own navigation).

The above requirements show that the user interface has to provide options for setting different modes, and several navigation options to support different kinds of histories in this location-aware system.

4 Interaction Design Using Discrete Event Modelling

While the message sequence diagrams developed in the previous section provide a good description of the requirements and possible usage scenarios of the TIP system, they cannot show the dynamics of the system behaviour and the complex interactions with the user interface. In contrast to modelling for an immobile desktop system, the small-screen interface on a mobile device considerably influences the system behaviour. Together with the necessary interactions with all the other services and the location-related aspects, this leads to very sophisticated concurrent interactions that need to be modelled carefully.

4.1 The Missing Link: Interfaces and States

Douglas (Douglas 1999) distinguishes between simple, state, or continuous behaviour of objects in systems modelling. We identify the dynamic behaviour of the services in the TIP system as state-full. Therefore, finite-state machines are an appropriate technique to model these concurrent interactions. UML supports the modelling of finite-state machines with UML statecharts and UML activity diagrams (Object Management Group 2003, Douglas 1999). While UML statecharts have been used successfully to design reactive and real-time systems, there still is only limited tool support for the formal analysis of complex statecharts models.

We strongly believe that, when modelling concurrent activities, the designer needs to be able to interact with the system to explore its capabilities and pitfalls. We therefore do not use UML statecharts, but a modelling tool for discrete event systems, which allows us to analyse the model for conflicts, and allows for interaction and playful exploration of the modelled system. The relationship between UML statecharts and discrete event systems, and a possible translation between the two formalisms is explored in (Malik & Mühlfeld 2003).

4.2 Principles of Discrete Event Modelling

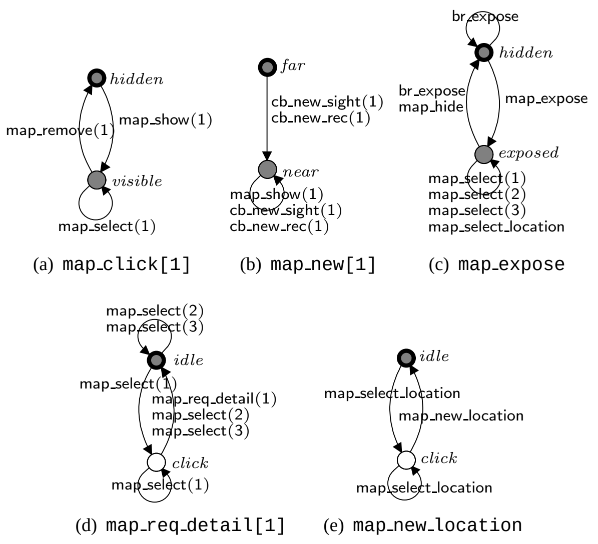
We use the VALID Toolset to model and analyse the behaviour of the TIP client. The VALID Toolset, developed at Siemens Corporate Technology, supports the modelling, verification, and code generation of finite-state automata as described in (Brandin, Malik & Malik 2004, Malik & Mühlfeld 2003). It uses the modelling methodology of discrete event systems (Cassandras & Lafortune 1999, Ramadge & Wonham 1989).

In this framework, concurrent systems are modelled using several *finite-state automata* running in parallel. Each automaton is represented graphically as a state transition graph as shown in Figure 8. States are represented as nodes, with the initial state highlighted by a thick border, and terminal states coloured grey. Transitions are represented as labelled edges connecting the states: in Figure 8(c), e.g., the automaton `map_expose` changes its state from $hidden$ to $exposed$ with an occurrence of the event map_expose. If an edge is marked with more than one label, this indicates that any one of the corresponding events causes the transition. Automaton `map_expose` in Figure 8(c), e.g., changes from state $exposed$ to state $hidden$ if a br_expose or a map_hide event occurs.

If a certain state does not have any outgoing transitions labelled with a certain event, that event is *disabled* by the automaton and cannot occur in the system. For example, automaton `map_expose` in Figure 8(c) disables event map_expose when in state $exposed$. This rule applies only to events that occur somewhere in an automaton or, more precisely, to events that constitute the so-called *event alphabet* of the automaton. Events that do not occur in the event alphabet remain enabled and do not cause any state change of the automaton when they occur. For example, event map_show(1) occurs in the event alphabet of `map_new[1]` in Figure 8(b), but not in `map_expose` in Figure 8(c). Therefore, automaton `map_expose` is always ready to perform a transition with event map_show(1), leaving its state unchanged.

In this way, several automata are composed by *synchronisation on common events*. All synchronised automata repeatedly agree on an event to be executed next, and simultaneously perform the corresponding state transition. A state transition using an event σ can only take place if all synchronised automata that have σ in their event alphabet allow the event σ to occur.

For more details on synchronous composition and other concepts from the theory of discrete event systems, please refer to (Cassandras & Lafortune 1999, Ramadge & Wonham 1989). The benefit of this very simple framework is that it does not only enable us to create and simulate a concise model of the dynamic system behaviour, but also makes it possible to use *model checking* (Clarke, Grumberg & Peled 1999). This ability to check critical properties of the model quickly and automatically has helped us to discover several problems.



(a) `map_click[1]` (b) `map_new[1]` (c) `map_expose`

(d) `map_req_detail[1]` (e) `map_new_location`

Figure 8: The map model

4.3 A Model of the TIP Client

Our model of the TIP system includes all the components constituting the TIP client on a mobile device as shown in Figure 2, namely the location service, the map service, the browser, and the client broker. It also includes a model of the behaviour of the event-based middleware (ENS), to the extent needed to capture the communication between the client and the ENS. The details of the information systems on the server side are not modelled.

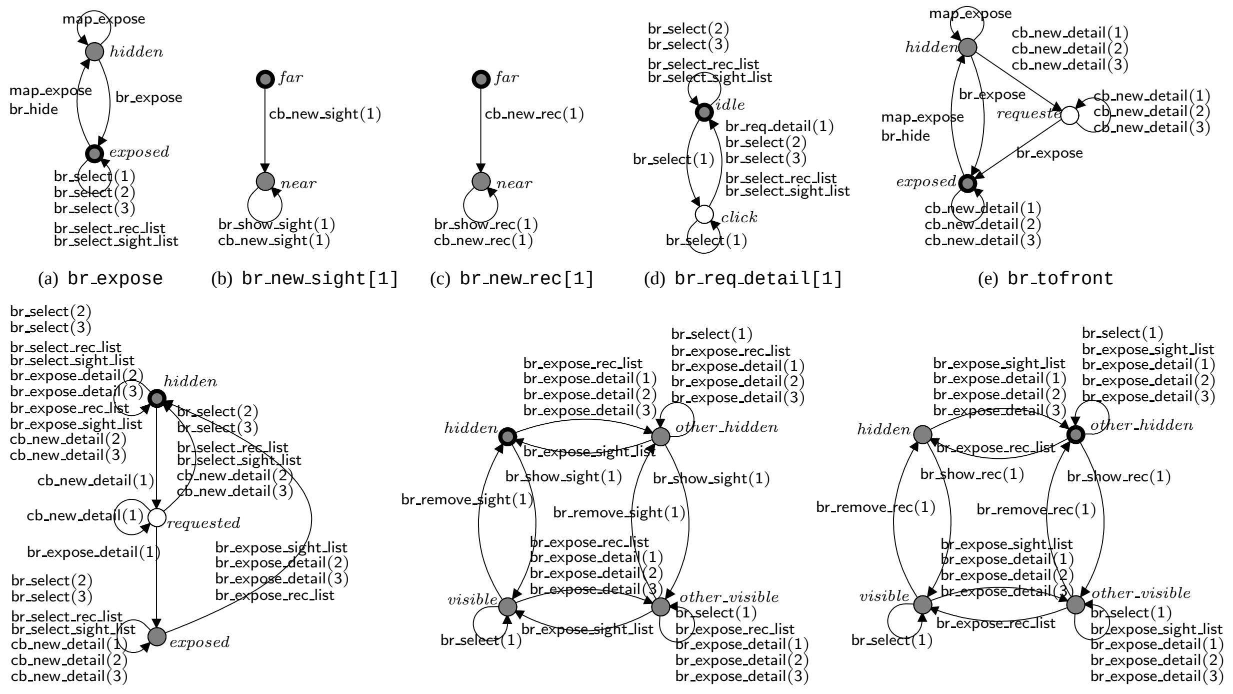
When modelling a system that includes large amounts of complex data, such as TIP, using discrete event systems, abstractions are necessary in order to guarantee that the state space remains finite. In a discrete event model, it usually is not possible to include details about the information attached to the various events that may occur—only the fact that a message is sent or received is modelled, in most cases completely abstracting away from the associated data.

In some cases, a certain reference to the data is needed to retain an interesting aspect of possible interactions. In our model, we assume that there are three different Sights 1, 2, and 3 that can be displayed. This number guarantees a tractable model, and is completely sufficient to reveal conceptual problems in the design. For clarity of the model, we also used other small simplifications.

In the following, we describe each of the service components separately. The next section explains the interplay of these components, and shows some of the results obtained from the formal analysis of the model. A more detailed model can be found in (Hinze, Malik & Malik 2005).

The Map Service. The map service shows the location of the user as well as sights and recommendations nearby. For the purpose of modelling the map service, sights and recommendations are treated alike, although they may be displayed differently on the screen. Figure 8 shows a simplified model of the map service, which only supports a single map with a fixed number of up to three sights or recommendations. Dynamic updates of the map area are not supported, nor is the removal of sight information, nor the highlighting of the sight shown in the browser.

Since the model is restricted to three sights, the map can show none, one, two, or three sights. Automaton `map_click[1]` in Figure 8(a) models whether Sight 1 is shown on the map or not. There exist similar automata for Sights 2 and 3, which are not depicted in Figure 8 since the only difference is the sight number in brackets.



(a) br_expose (b) br_new_sight[1] (c) br_new_rec[1] (d) br_req_detail[1] (e) br_tofront

(f) br_expose_sight[1] (g) br_click_sight[1] (h) br_click_rec[1]

Figure 9: The browser model

Initially, Sight 1 is not shown. Becoming visible is modelled by event map_show(1), and becoming invisible is modelled by event map_remove(1), both changing the state of automaton map_click[1]. The automaton uses another event, map_select(1), which does not change the state of the automaton. The important point here is that event map_select(1) is only possible when Sight 1 is shown on the map; an obvious consequence of how users can interact with their device.

The map highlights known sights and makes them selectable for the user. Automaton map_new[1] in Figure 8(b) models that Sight 1 is shown on the map after the map has received information about it as a sight (event cb_new_sight(1)) or as a recommendation (event cb_new_rec(1)) from the client broker. Again, there are similar automata for Sights 2 and 3 not depicted here. Also note that automata map_new[1] and map_click[1] synchronise on the common event map_show(1), i.e, the event map_show(1) is only possible if both automata are in a state where this event is allowed.

There are more factors that influence whether it is possible to click at the map browser. The map can be either exposed, which means it is visible on the screen, or hidden, and thus not visible on the screen. The user is only able to click onto the map if it is exposed. This behaviour is modelled by automaton map_expose in Figure 8(c).

Automata map_req_detail[1] in Figure 8(d) and map_new_location in Figure 8(e) describe the behaviour of the map if the user clicks on it. The user can select sights displayed on the map to request for detailed information, or click elsewhere to explore other areas. After one or possibly more clicks on Sight 1, the map sends a map_req_detail(1) event to the client broker, requesting detailed information to be shown in the browser. The attempt to send this message is cancelled if the user clicks on another sight. If the user clicks on the map to browse to a different location (event map_select_location), a map_new_location event is sent to the client broker in order to start searching for sights and recommendations close to the selected location.

The Browser. The browser is used to show textual information about sights visited by the user. In the model, it can display a list of sights close to the current position of the user, a list of recommendations close to the current position of the user, or the detailed information of a specific sight. It can display information in the form of web pages received from the client broker, and request detail information, if the user clicks on hyperlinks. The current model assumes an intelligent browser that can combine information from several messages and change the display accordingly.

When a sight or recommendation gets close enough, the browser receives a cb_new_rec(1) or cb_new_sight(1) message from the client broker. In response to this, the browser updates its sight or recommendation list, indicated by event br_show_rec(1) or br_show_sight(1), as modelled by automata br_new_rec[1] in Figure 9(c) and br_new_sight[1] in Figure 9(b).

If the user clicks on a hyperlink to request details about Recommendation or Sight 1 (event br_select(1)), a br_req_detail(1) event is sent to the client broker unless the user clicks onto another recommendation or sight before that event could be sent. This is modelled in automaton br_req_detail in Figure 9(d) and works like the processing of clicks on the map. The client broker responds to the br_req_detail(1) event by sending a cb_new_detail(1) event, providing a web page with detailed information about the sight. As soon as this message arrives, the browser is exposed (unless it is already, see automaton br_tofront in Figure 9(e)), and the information about the sight is displayed (see automaton br_expose_sight in Figure 9(f)).

A more detailed description of all automata in Figure 9 is given in (Hinze et al. 2005).

The Client Broker. The client broker links the applications on the mobile device with the network and the event notification system (ENS) on the server side. It also coordinates the map display and browser to ensure that control is passed correctly between these two applications.

In most cases, the client broker simply forwards events received from the applications to the ENS and vice versa.

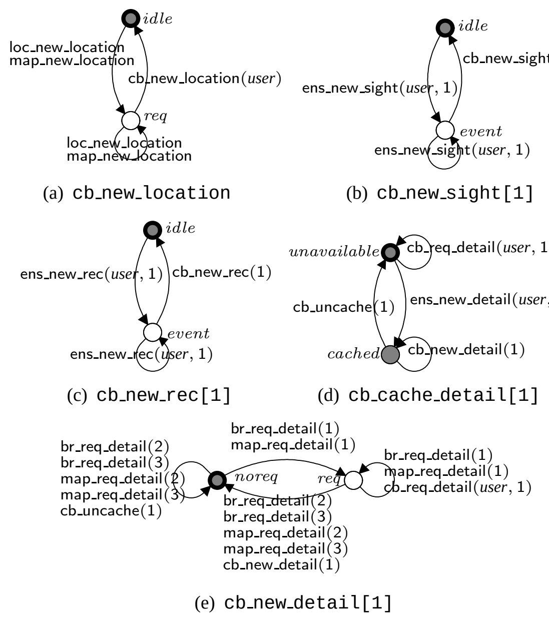


Figure 10: The client broker model

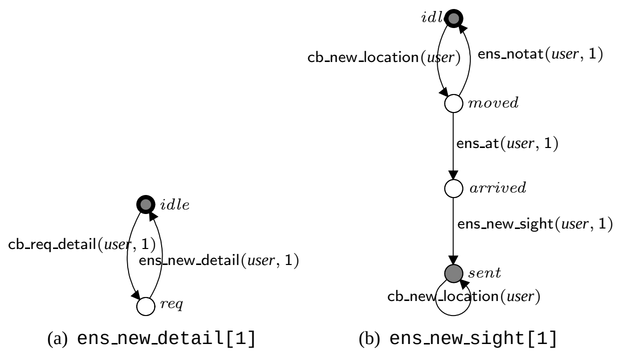


Figure 11: The ENS model

For example, automaton `cb_new_location` in Figure 10(a) forwards location changes from the client's location server (event loc_new_location) or from the map (event map_new_location) to the ENS after attaching the user ID. Likewise, automata `cb_new_sight` in Figure 10(b) and `cb_new_rec` in Figure 10(c) model how information from the ENS about new sights or recommendations, respectively, is sent back to the map and browser.

The model for detailed sight information, as given by automata `cb_new_detail[1]` in Figure 10(e) and `cb_cache_detail[1]` in Figure 10(d), is more elaborate because the client broker also has the ability to cache some of the web pages received. The browser or map can request detailed information about Sight 1 by sending a br_req_detail(1) or map_req_detail(1) event, which causes the automaton `cb_new_detail[1]` to enter state *req*. In this state, the client broker is ready to forward the request augmented with the user ID to the ENS (event cb_req_detail$(user, 1)$). This event is answered by the ENS by providing detailed information (event ens_new_detail$(user, 1)$), which is stored in the cache, indicated by automaton `cb_cache_detail[1]` changing into state *cached*. At this stage, the client broker can send a cb_new_detail(1) event to the browser, which will display the detailed information.

The Server Side. The model of the server side states the assumption that all requests sent to the ENS are actually answered. Automaton `ens_new_detail` in Figure 11(a) shows that a request for detailed information in form of a cb_req_detail$(user, 1)$ event is answered by an ens_new_detail$(user, 1)$ event with the corresponding web page. Furthermore, the information servers respond to location changes of the user, signalled by cb_new_location$(user)$ events. If such an event occurs, the databases are searched for sights that are close to the new position. The result of the search is modelled by the two events ens_notat$(user, 1)$ and ens_at$(user, 1)$. If Sight 1 is close enough (event ens_at$(user, 1)$), this information is sent as an ens_new_sight$(user, 1)$ event to the client broker. Similar automata are used for recommendations.

Table 1: Example trace corresponding to Figure 3

	`cb_new_location`	`ens_new_sight[1]`	`cb_new_sight[1]`	`map_new[1]`	`br_new_sight[1]`
(initial state)	*idle*	*idle*	*idle*	*far*	*far*
loc_new_location	*req*	*idle*	*idle*	*far*	*far*
cb_new_location$(user)$	*idle*	*moved*	*idle*	*far*	*far*
ens_at$(user, 1)$	*idle*	*arrived*	*idle*	*far*	*far*
ens_new_sight$(user, 1)$	*idle*	*sent*	*event*	*far*	*far*
cb_new_sight(1)	*idle*	*sent*	*idle*	*near*	*near*
map_show(1)	*idle*	*sent*	*idle*	*near*	*near*
br_show_sight(1)	*idle*	*sent*	*idle*	*near*	*near*

4.4 Analysing the Model

Having created the model, it is possible to run simulations and check whether the model can perform the interactions specified by the sequence diagrams. For example, Table 1 shows the event sequence corresponding to the basic interactions following a location change as specified in Figure 3.

When a location change occurs, the location server on the mobile device sends a loc_new_location event to the client broker. This event triggers the automaton `cb_new_location` to send a cb_new_location$(user)$ event to the ENS. The ENS, in this case represented by automaton `ens_new_sight`, may now detect that the user is close to Sight 1 (event ens_at$(user, 1)$) and respond by sending an ens_new_sight$(user, 1)$ event. This message in turn triggers automaton `cb_new_sight`, which forwards the event as cb_new_sight(1) to the map service and browser. The cb_new_sight(1) event simultaneously activates automaton `map_new` in the map service and `br_new_sight` in the browser, which causes them to display the new sights by events map_show(1) and br_show_sight(1), respectively.

In the discrete event model, a single cb_new_sight(1) from the client triggers actions both by the map service and the browser. The same communication is modelled by two separate interactions in the message sequence diagram in Figure 3. The notion of the client broker sending a single event, without necessarily knowing the receivers, seems to capture the intended behaviour of the event-based middleware more accurately.

In addition to simulating the possible behaviours of the model, it is possible to perform some formal analysis and check whether the model satisfies properties of interest. We have checked that the model is *controllable* (Cassandras & Lafortune 1999, Ramadge & Wonham 1989), i.e., that it is always ready to react to any possible events caused by user interactions or the information systems, and that it is *nonblocking* (Cassandras &

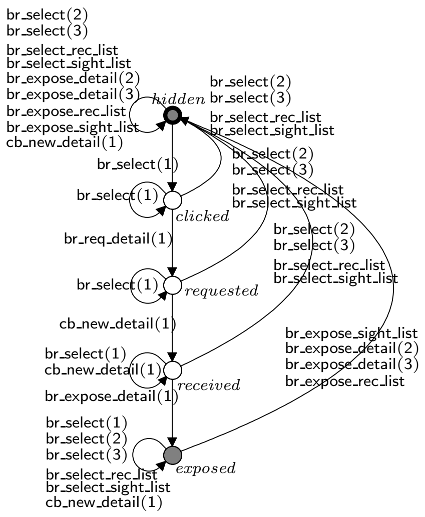


Figure 12: Broken version of `br_expose_sight[1]`

Lafortune 1999, Ramadge & Wonham 1989), i.e., that it is free from deadlocks and livelocks.

These checks did not succeed immediately. By repeatedly verifying properties, we discovered several problems in the model. Then we eliminated the problems and checked the model again until all critical properties could be checked.

An earlier version of the model included the automaton in Figure 12. It represents an attempt at creating a smarter browser that tries to avoid showing sight details if they are not the last thing the user clicked on. The idea is that, if the user clicks a link to Sight 1 in the browser, the browser requests detailed information about this sight from the client broker. However, this request is cancelled if the user clicks on something else before processing can be completed.

Formal analysis shows that this approach does not work. If we replace automata `br_req_detail[1]` in Figure 9(d) and `br_expose_sight[1]` in Figure 9(f) by the alternative in Figure 12, and make similar changes for Sights 2 and 3, the model contains a deadlock and therefore cannot be proven to be nonconflicting. When trying to verify this property, the VALID Toolset produces a counterexample that reveals a problem in the design.

The problem is as follows. The user may click on a link to Sight 1 (event br_select(1)), which causes the browser to send a request for details about Sight 1 to the client broker (event br_req_detail(1)). At this stage, automaton `br_expose_sight[1]` in Figure 12 is in state *requested* and waits for the client broker to send detail information about Sight 1. But before this information arrives, the user may activate the map (event map_expose) and click on sight two on the map (event map_select(2)). As a result, the map server sends a request for details about sight two to the client broker (event map_req_detail(2), see Figure 8(d)). This causes the client broker to cancel the request for details about Sight 1, and only to process the second request (see Figure 10(e)). As a result, the browser will never receive information about Sight 1, i.e., the automaton in Figure 12 will never receive the cb_new_detail(1) event needed to leave state *requested*.

The root of this problem is that the browser cannot accurately determine which is the most recent request from the user. The browser only knows about user interactions with the browser, not with the map service. In consequence, the decision about the most recent request must be shared between the two applications. To overcome the problem, the model has been changed to the one presented in Figure 9, where the client broker determines which request is the most recent by checking the events received from the browser and map service. The browser merely displays all the detail pages received.

After fixing all problems, the model has been verified successfully by the VALID model checker. All checks can be completed in three minutes on a 1.4 GHz Athlon processor with 256 MB of RAM. This shows that the finite-state automata model already is fairly complex and serves as a challenging benchmark for model checking tools.

5 Discussion of Related Work

This work touches on the research of the systems design and HCI communities. We also consider typical approaches for modelling and evaluating location-aware mobile systems.

System Modelling and Design. Modelling of concurrent systems using state machines has chiefly been used for mission critical applications, often focusing on safety issues, e.g., flight interface control (Degani, Heymann, Meyer & Shafto 2000). Typical problems that have also been identified in our study are mode awareness, mode confusion, and automation surprise. Bolignano et al. (Bolignano, Le Métayer & Loiseaux 2000) call the application of formal methods in practical applications a 'missing link'.

For the design of interaction and interfaces for virtual environments, CSP (Communicating Sequential Processes) has been used (Smith, Marsh, Duke, Harrison & Wright 1998, van Schooten, Donk & Zwiers 1999). CSP is an alternative modelling principle which we could have used for our system. CSP is related to our modelling approach, but typically lacks modelling environments that support active simulation. In CSP applications, the aspects of location-awareness and mobile users are rarely considered. In virtual environments (as the ones mentioned above), the mobility of the user's avatar does not interfere with the location of the real user as is the case in our application.

User and Interaction Modelling. HCI research has focused on modelling of users and their interaction with a system. Blandford et al. evaluate different programmable user modelling (PUM) strategies for user-centred criteria such as tractability and re-usability (Blandford, Butterworth & Curzon 2004). Within their schema of operational and abstract models, our approach follows an abstract functional model, with the main advantage of provability. (Thimbleby, Cairns & Jones 2001) proposes modelling push-button devices using Markov models; this approach is similar to ours but evaluates a different kind of application. Degani et al. (Degani, Shafto & Kirlik 1999) address the problem of mode confusion using statecharts and modelling structures; their findings are more concerned with user interface design. Their modelling focuses on concurrency issues. The problem of mode confusion has also been addressed using formal modelling methods (Rushby 2002, Butler, Miller, Potts & Carreño 1998). These approaches chiefly deal with the awareness of the users of different modes (focusing on opacity, complexity, incorrect mental model) and on safety (one of the typical application fields of formal methods). The focus of the cited works is different to ours; in their evaluations, interactions between system components are not considered, nor are concurrency and synchronisation problems. In addition, the issues of location and user movements introduce challenges that are not addressed.

Design and Evaluation of Location-Aware Mobile Systems. Typical design and evaluation uses methods from desk-bound computers (Kjeldskov et al. 2005). The focus lies on the interface design of small screen devices and user-device interactions. Interaction design that takes location-awareness and mobility into account is rare.

In the distinction between user-centred and technology-centred design (Kjeldskov & Howard 2004, Brown 1998), our approach is closer to technology-centred design. We use reflection about an existing prototype and technology-driven service models, enriched with usage scenarios and enactment of future scenarios. The typical user-centred approach is followed in (Pousman, Iachello, Fithian, Moghazy & Stasko 2004): questionnaires, interviews and discussion with potential users lead to system design with subsequent user-testing of a functional prototype. The findings of this project underline that design for mobile hand-held devices needs to be distinct from design for desktop machines. One result is the necessity to easily restart or resume actions that have been interrupted. We address this issue by the introduction of different navigation methods in Section 3.3, which give users easy access to their current and previous states within the system interaction, and by actively propagating context changes to all services. Our design addresses selected issues (interaction syntax, resuming tasks, integration) identified in (Pousman et al. 2004) in an a-priori manner. Early HCI considerations for location-aware applications have been discussed in (Brown 1998); in our redesign process, we identify and address similar issues.

A-posteriori Usability Studies. Usability and interaction issues of mobile location-based systems have often been evaluated a-posteriori, that is after the implementation of the software (Iacucci, Kela & Pehkonen 2004). Kjeldskov et al. (Kjeldskov et al. 2005) point out that the special characteristics of tourist guides make the design and evaluation a challenging task: close relation to physical location and objects in immediate user surroundings as well as the ambulating user constantly changing physical location. They identified as critical usability issues those related to the mapping issues from the use of the system in the real world, such as (1) representation of information and interaction between user and system depending on the surrounding environment and (2) disparities in the relationship presented in the system and the context in which the user was situated. These are two of the issues that we also identified as critical in our design phase. Thus, critical issues typically observed in usability studies could be addressed already in the design phase.

Summary. Traditionally, formal modelling has mainly been used to address issues of concurrency, safety, and correctness. Mobility and location-awareness usually is not addressed directly. When addressing interaction design for mobile systems, typically only the aspects of user interaction are considered, since this community mainly regards problems of interactions using small screen displays. Location-aware mobile applications pose more challenges such as identity of locations, virtual/real-world confusion, complex interactions with cooperating services on the same device. When designing mobile location-aware applications, scenario and prototype-based approaches prevail. Only a few publications have dealt with the modelling of location-aware systems. Most evaluations are a-posteriori usability studies.

6 Lessons Learned

From our experience with modelling the system, we have identified the following issues. A clear distinction is needed between three user identities:

1. *The real-world user.* This is the person using the system; they can change their location in the real world. This location is captured by the location service in the system and triggers various responses.
2. *The virtual user.* This is the representation of the real-world user within the system. Typically the virtual user follows the movements of the real world user. The virtual user can also be used to 'visit' places within the system on behalf of the user (e.g., by clicking on the map to explore places at a distance). It should be possible to move through the location history of the virtual and the real user (e.g., using user-defined location points).
3. *The interacting user.* Independent of the location of the user, the interface allows to select browser pages, pictures and maps. The interaction is controlled by the real-world user. It should be possible to move through the interaction history with the system (similar to the 'Back' and 'Forward' buttons in a browser).

Several issues for the design of interactions between services, and between user and system have been identified. These issues occur as an immediate consequence of the mobility and location awareness on the system. They are typical for this kind of system.

The first one is mode confusion regarding the different users and places (e.g., location of the real user on the map while browsing with the virtual user). The user interface has to address this issue in a clear and comprehensible manner. This is especially important for the navigation through interaction and location histories. On the other hand, the server and involved services have to address this issue as well. For example, for deciding whether to include a certain location into the user history, whether to allow the user to give feedback about a place, or which information to display.

The second issue is location capturing. Several types of location capturing are supported for mobile systems; all provide varying semantics of new-location data (e.g., location measured after a time interval or depending on movement). These different types influence how the system can react, and how location events should be handled. A related issue is that of user movement in interaction with location capturing. This may lead to confusion about the actual location of the real-world user.

A third observation is that the context of the user has to be carried as explicit information, that is, the system has to react differently depending on its current context. This has been expressed earlier as 'modes', but that concept does not go far enough. Context is more complex than simple modes. Concepts from event-based systems with states may be explored here.

The fourth issue was identified when reasoning about the resulting model: System components may themselves have to be distributed (i.e., parts of the component reside on the client and other parts on the server). The current model assumes a certain part of the programming logic to reside on the client device (e.g., to change the display modes depending on the context). This may not be feasible for third-party thin clients such as standard web browsers displaying information from server-side JSP pages. This creates a major problem for re-usability of existing services and interfaces.

7 Conclusions

This paper presents the lessons learned during the redesign of a mobile location-aware system, using a combination of UML message sequence diagrams and discrete event systems. It proposes an improved service-oriented architecture and points out that different contexts need to be distinguished with great care in mobile location-aware systems, with implications not only for the user interface but also for the interaction protocols between the various services involved.

The formal modelling approach has greatly improved the system design. By describing the requirements formally, developers are forced to think carefully about the planned system. The activity of specifying and modelling on its own helps to understand and clarify many aspects of the system to be constructed.

We have seen that UML message sequence diagrams are very useful to describe particular sequences of actions, but fail to reveal problems that occur when several such action sequences are running together at the same time. Therefore, a finite-state automata model of the system has been created and analysed. The result is an abstract model of the *behaviour* of the system, which makes it possible to simulate the interaction between the various components and gain a thorough understanding of the dynamics of the system.

Exhaustive simulation and analysis of the model allows for finding subtle problems at this very early stage of the design. Several issues with the proposed interaction protocols have been discovered and solved, which may otherwise have remained undetected until much later during the implementation of the system.

As a next step for the interaction design, we plan to address the issues of interaction with other mobile peers and distribution of system components. For the overall system design, we plan to (1) support more services to identify common abstractions for a service oriented architecture; (2) include trust-based information exchange; and (3) model different applications with different services but the same base architecture. For the design process using discrete event systems, we plan to implement an interface simulation to allow for more direct user interaction with the simulated system.

References

Blandford, A., Butterworth, R. & Curzon, P. (2004), ‘Models of interactive systems: a case study on programmable user modelling’, *Int. J. Hum.-Comput. Stud.* **60**(2), 149–200.

Bolignano, D., Le Métayer, D. & Loiseaux, C. (2000), Formal methods in practice: the missing link. a perspective from the security area, *in* ‘Modeling and Verification of Parallel Processes (MOVEP 2000)’.

Brandin, B. A., Malik, R. & Malik, P. (2004), ‘Incremental verification and synthesis of discrete-event systems guided by counter-examples’, *IEEE Trans. Contr. Syst. Technol.* **12**(3), 387–401.

Brown, P. (1998), Some lessons for location-aware applications, *in* ‘Proc. 1st Workshop on HCI for Mobile Devices’, pp. 58–63.

Butler, R., Miller, S., Potts, J. & Carreño, V. A. (1998), A formal methods approach to the analysis of mode confusion, *in* ‘AIAA/IEEE Digital Avionics Systems Conf.’.

Cassandras, C. G. & Lafortune, S. (1999), *Introduction to Discrete Event Systems*, Kluwer.

Clarke, Jr., E. M., Grumberg, O. & Peled, D. A. (1999), *Model Checking*, MIT Press.

Degani, A., Heymann, M., Meyer, G. & Shafto, M. (2000), Some formal aspects of human-automation interaction, Technical Report NASA/TM-2000-209600, NASA Ames Research Center, Moffett Field, CA.

Degani, A., Shafto, M. & Kirlik, A. (1999), ‘Modes in human-machine systems: Constructs, representation, and classification’, *Int. J. Aviation Psychology* **9**(1), 125–138.

Douglas, B. P. (1999), ‘UML statecharts’, *Embedded Systems Programming* **12**(1).

Hinze, A. & Buchanan, G. (2005), Context-awareness in Mobile Tourist Information Systems: Challenges for User Interaction, *in* ‘Proc. Workshop on Context in Mobile HCI, in conjunction with Mobile HCI’, Salzburg, Austria.

Hinze, A. & Buchanan, G. (2006), The challenge of creating cooperating mobile services: Experiences and lessons learned, *in* ‘Proc. 29th Australasian Computer Science Conf. (ACSC 2006)’, Hobart, Australia. To appear.

Hinze, A. & Junmanee, S. (2005), Providing recommendations in a mobile tourist information system, *in* ‘Information Systems Technology and its Applications, 4th Int. Conf. (ISTA 2005)’, Palmerston North, New Zealand.

Hinze, A., Malik, P. & Malik, R. (2005), Towards a TIP 3.0 service-oriented architecture: Interaction design, Technical Report 08/05, Dept. of Computer Science, University of Waikato.

Hinze, A. & Voisard, A. (2003), Location- and time-based information delivery in tourism, *in* ‘Advances in Spatial and Temporal Databases (SSTD 2003)’, Vol. 2750 of *LNCS*, Satorini Island, Greece.

Iacucci, G., Kela, J. & Pehkonen, P. (2004), ‘Computational support to record and re-experience visits’, *Personal Ubiquitous Comput.* **8**(2), 100–109.

Kent, S. (2001), *Formal methods for distributed processing: a survey of object-oriented approaches*, Cambridge University Press, New York, NY, USA, chapter The unified modeling language, pp. 126–152.

Kjeldskov, J., Graham, C., Pedell, S., Vetere, F., Howard, S., Balbo, S. & Davies, J. (2005), ‘Evaluating the usability of a mobile guide: The influence of location, participants and resources’, *Behaviour and Information Technology* **24**(1), 51–65.

Kjeldskov, J. & Howard, S. (2004), Envisioning mobile information services: Combining user- and technology-centered design., *in* ‘Proc. Asia-Pacific Conf. Human-Computer Interaction (APCHI 2004)’.

Malik, R. & Mühlfeld, R. (2003), ‘A case study in verification of UML statecharts: the PROFIsafe protocol’, *J. Universal Computer Science* **9**(2), 138–151.

Object Management Group (2003), ‘Unified modelling language (UML), version 1.5’. Available at `http://www.omg.org`.

Pousman, Z., Iachello, G., Fithian, R., Moghazy, J. & Stasko, J. (2004), ‘Design iterations for a location-aware event planner’, *Personal Ubiquitous Comput.* **8**(2), 117–125.

Ramadge, P. J. G. & Wonham, W. M. (1989), ‘The control of discrete event systems’, *Proc. IEEE* **77**(1), 81–98.

Rushby, J. (2002), ‘Using model checking to help discover mode confusions and other automation surprises’, *Reliability Engineering and System Safety* **75**(2), 167–177.

Smith, S., Marsh, T., Duke, D., Harrison, M. & Wright, P. (1998), Modelling interaction in virtual environments, *in* ‘Proc. UK-VRSIG ’98’, Exeter, UK.

Thimbleby, H., Cairns, P. & Jones, M. (2001), ‘Usability analysis with markov models’, *ACM Trans. Comput.-Hum. Interact.* **8**(2), 99–132.

van Schooten, B., Donk, O. & Zwiers, J. (1999), Modelling interaction in virtual environments using process algebra, *in* ‘Proc. 15th Twente Workshop on Language Technology’.