

# **An Integrating Framework for Electronic Aids to Support Journeys by Visually Impaired People**

Peter Green and Simon Harper

## *Abstract*

*A framework for integrating current and future mobility devices into a system to support navigation around an urban environment by a visually impaired traveller is presented, based on a detailed analysis of the travel task. An executable model of the framework is discussed, and progress in validating the model's behaviour is presented. Implementation options are also briefly considered.*

## **1.Introduction**

Basic independent mobility is vital throughout all sections of society. However, urban environments are overwhelmingly designed to support travel by sighted individuals, and the needs of the visually impaired are not considered [4]. Technology can be employed to support travel by visually impaired individuals, but in doing so it is important to understand the overall nature of the travel task, and to develop systems that provide appropriate support. The work reported here, and in [3], is an effort to establish a framework for mobility/orientation systems, based upon an analysis of the travel task, and on typical mobility/orientation aids. The approach that has been adopted has been based upon a number of observations, discussed in detail in [2] and briefly outlined below :

- The travel task is complex and composed of a number of sub-activities, each requiring different forms of support.
- The environment must be able to communicate information about itself by mechanisms that are readily accessible to the visually impaired traveller.
- Many travel aids have been proposed, but most support only one sub-activity of the travel task. Hence several aids are needed for a typical journey, if full support is to be provided.
- Carrying and using several travel aids will be at best awkward, and at worst infeasible.

We concluded that what is needed is a portable device that fuses data from several aids, both ‘on-board’ and in the environment, and which supports all aspects of the travel task. A key issue is to hide the details of individual mobility aids, hence providing high level support for travel.

The objective of this work is to develop a framework (POLI - Portable Orientation and Location Interface) for systems that integrate a number of devices and provide support for independent travel by visually impaired people. Evaluation of such a framework by construction and testing is, in the first instance, unnecessarily costly and time-consuming. Hence our approach has been to develop a fully executable system model, using the MOOSE method (Section 3), which provides a well-defined route for transforming executable models into full system implementations. Hence the model presented in this paper may be viewed either as a simulation, or as a partial implementation.

## 2.The Travel Task

As discussed in [3], the travel task consists of a number of simpler activities:

- **Route Planning:** The journey is planned, and a route is chosen beforehand, based on maps and/or previous knowledge of the route or journey.
- **Obstacle Detection/Avoidance:** Moving and stationary objects must be avoided during the journey.
- **Orientation and Waypoints:** Journeys may be divided into sections based on waypoints and way-edges. Waypoints are elements of the environment that enable travellers to orientate themselves. Way-edges are a generalisation of waypoints, and describe large-scale objects that may be followed for a period during a journey.
- **Use of Information Points:** Information points are points where information about the journey is available (timetables, next bus information, etc).
- **In-Route Guidance:** This may be performed by requesting directions or by carrying some form of map.

## 3.The POLI Framework

The framework describes how a number of mobility devices/travel aids can be integrated to enable a visually impaired user to interact with just one device to derive information from several sources in the environment. To accomplish this, the system must provide a mechanism for interacting with the

devices in the environment, and facilitate both one and two way information flows. It must also enable information to be acquired from remote systems. Based upon these observations, a consideration of the travel task and the characteristics of existing mobility aids, it is clear that in general, a POLI system must contain or interact with a number of generic device types [2]:

- **Hand-held Receivers:** This is the device carried by the user, which in addition to receiving information, facilitates user-interaction, and supports the entry of Route Planning information.
- **Standalone Devices:** These provide one way transmission from the environment and contain little intelligence themselves. Devices marking waypoints could be an example of this category e.g. a Sound Buoy [1].
- **Gateway Devices:** These require a two-way interaction, for example information points such as electronic timetables.
- **Repository Devices:** These notify hand-held receivers that a large amount of complex information is required from elsewhere, possibly by a wireless Ethernet, pager, or mobile phone link to augment In-Route Guidance.

POLI has been developed to accommodate multiple devices from each category. It provides a detailed architecture describing the interactions between the system and the devices, and a model of how data from the environment may be combined to provide user support.

The hand-held receiver was originally envisaged to be a custom embedded system attached to a PDA (Personal Digital Assistant), which would provide interfaces with mobility aids and device management, and also support output modalities not normally associated with PDAs (e.g. speech). However, as will be seen below, one of the advantages of using the MOOSE approach is that the system model is largely independent of the final implementation platform, and hence other implementation options can be accommodated.

#### **4.MOOSE - Model-based Object Oriented Systems Engineering**

MOOSE was selected for developing POLI because of its support for executable modelling and the transformation of models into implementations. Hence an executable model of a system can be developed and evaluated without the need to build a physical prototype. The model can then be transformed into an implementation, thus capitalising on the effort required to develop the model.

MOOSE is an Object Oriented (OO) approach to embedded system development, providing a lifecycle, a notation (graphical and textual), and a set of development heuristics. Embedded systems often require the development of application-specific software and hardware, and the system execution environment, and MOOSE supports all of these activities via object modelling. This capability is important for POLI since both hardware and software development will be necessary.

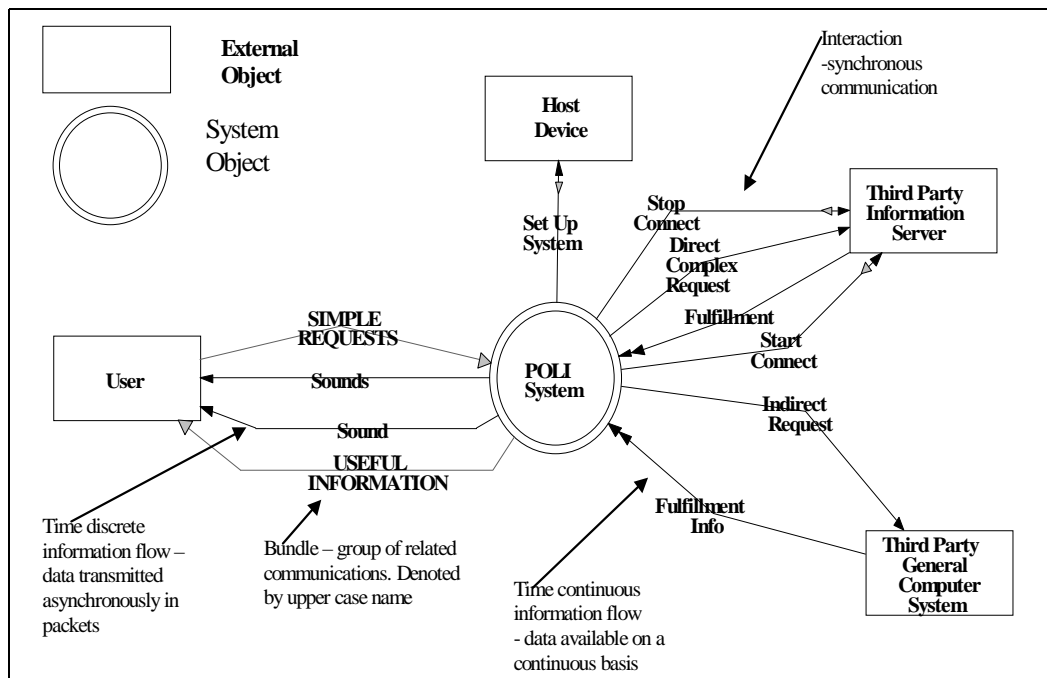
In the early stages of MOOSE, a hierarchical, graphical object model (the *Behavioural Model*) is developed based on the system's functional requirements. Objects are refined into lower level objects that provide the functionality of their parent. Lowest level objects are *primitive* and are not further decomposed. At this stage objects are *uncommitted* to hardware or software implementation. The behaviours of the primitive objects are specified in the next stage, both graphically, and using C++. This renders the model executable so that its behaviour can be validated against the functional requirements. In addition, it provides specifications of object implementations for possible use in later stages. The extended model is known as the *Executable Model*, and is tested against a number of usage scenarios [6]. When its behaviour is satisfactory, it forms an architecture that satisfies the functional requirements of the system. Since the model is uncommitted, it admits a number of different implementation options, and hence represents a system framework. The development of the POLI framework has been taken as far as the creation and testing of an executable model.

The later stages of the method involve 'committing' objects to implementations in software or hardware on the basis of non-functional considerations such as performance. The execution environment is then selected or developed. See [6] for further details.

## **5.The POLI System Model**

A complete MOOSE executable model of the POLI system has been developed, and tested by execution. In this section we present a highly elided version of the model. The top level of the

MOOSE object hierarchy is the External View (*Figure 1*), showing how the whole system (represented as POLI System) interacts with external objects beyond the system itself. Note that POLI System includes POLI devices located in the environment.



**Figure 1: External view of POLI and MOOSE notation**

*Figure 1* defines the interface between the system and its environment at an abstract level. This enables the design of the interface itself to be part of the overall system development. The Host Device object represents the basic hand-held device that the User carries, and with which the POLI system interfaces. The Third Party General Computer System object depicts a remote computer system that is one side of an information gateway. The other side of the gateway lies within the POLI System object itself, and controls communication with the Third Party General Computer System. The user communicates with the remote system by sending Indirect Request packets and waiting for Fulfillment Info from the remote system. This type of interaction will allow time-tabling systems etc. to be accessed. The Third Party Information Server represents a remote system that provides information based on requests from POLI. This is an instance of a Repository Device (see Section

3). This type of external differs from a gateway device, in that gateway devices typically facilitate two-way communication, whereas only a single request for information is made to a Third Party Information Server. A standalone device in the environment will communicate the existence of such a system to POLI transparently via a wireless link. To access the server a connection is made (Start Connection), and a Direct Complex Request is sent. The server will then fulfil this request and send Fulfillment information back. The connection will then be terminated (Stop Connection).

MOOSE is based on hierarchical refinement of complex objects into sets of simpler objects that together collaborate to provide the functionality of the 'parent'. Refining the POLI object from *Figure 1* yields *Figure 2*. The User Portable Device contains all the POLI functionality carried by the user, and controls input from the other two types of POLI device (Gateway POLI and Standalone POLI) that are located in the environment. It is also responsible for connecting to Third Party Information Servers if the Standalone POLI devices indicate the existence of such facilities.

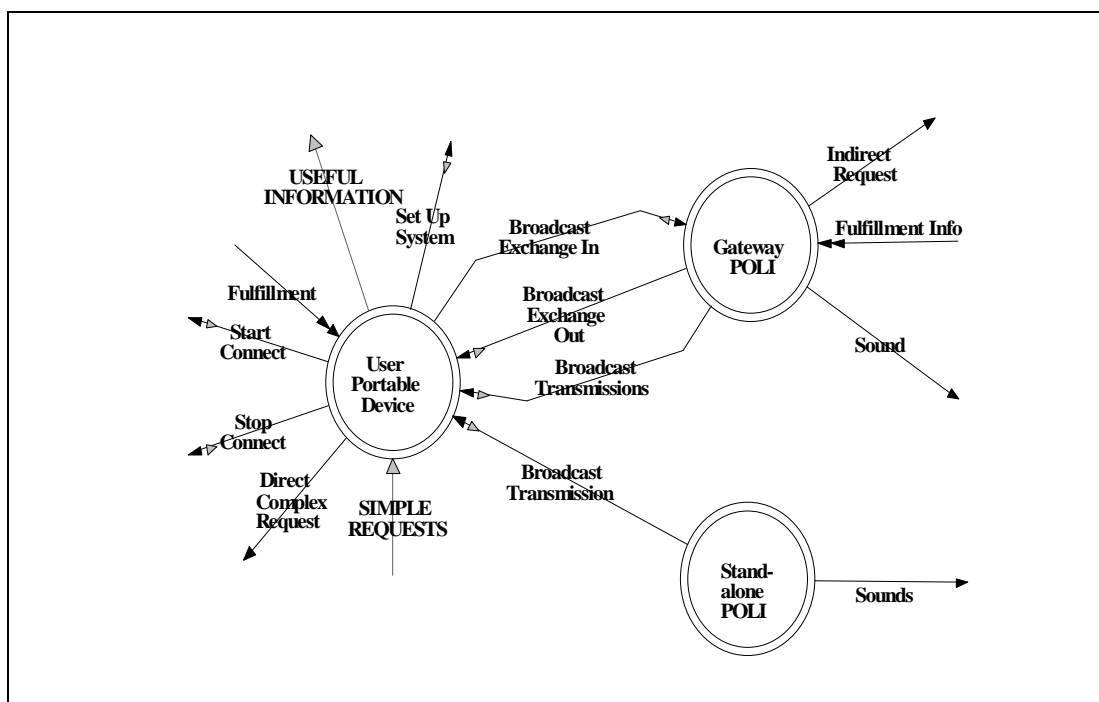


Figure 2: Refinement of POLI System

The Standalone POLI object represents a device that provides one-way output to the user. Sounds are emitted so the device can be located by orientating to the sounds emitted, and at the same time Broadcast Transmission(s) are made such that information about the device and its travel information can be accessed by the user. These devices will typically be waypoint or obstacle detection/avoidance devices.

The Gateway POLI object handles the data communications between itself and the Third Party General Computer System. When the Broadcast Transmissions arrive at the User Portable Device, (a Sound is also emitted to aid location) the user can connect to the gateway if they so wish. Then they can send a Broadcast Exchange In (passed on by the gateway), and wait for a Broadcast Exchange Out (supplied by the gateway). This is the case, for example, with Information points.

Figures 1 and 2 represent the highest levels of the MOOSE Behavioural Model of the POLI system. Each of the objects in Figure 2 has been further decomposed, and the behaviour of the primitive objects has been specified in C++. The resulting Executable Model has been synthesised into an C++ program using the MOOSE toolset. See [2] for further details.

## **6.Current Status and Conclusions**

A full executable model of the POLI framework has been synthesised and tested, and it currently exhibits the behaviour implicit in the description of Section 5. It is possible to run the model for specific journeys and to observe the behaviour of the system. Details of model execution for a specific journey, including screenshots from the MOOSE model animation tool, can be found in [2]. These are not reproduced here due to space constraints.

In order to move the executable POLI model towards an implementation, the objects within the model must first be committed to hardware or software implementation on the basis of issues such

as performance and cost. Preliminary considerations of this type are presented in [2]. The system execution environment must then be chosen. As indicated above, this was originally envisaged to take the form of a custom embedded system attached to a PDA. Infra-red communication was believed as the most attractive option for communication between standalone POLI devices and the hand-held device. Some form of docking was foreseen for communication with gateway devices. However, since the model was completed, a number of technological developments have changed the picture somewhat. These include the wireless applications protocol (WAP) [5], which facilitates internet access from mobile phones, and the Bluetooth specification for cheap, short-range radio links between mobile and stationary devices [4]. Both technologies look set to become de facto standards, and can provide an implementation platform for POLI based largely on mainstream technology. A key advantage of the MOOSE approach is that, since the executable model is independent of implementation technology, it is equally relevant to the newer implementation environment, and the effort to create the executable model can also be recouped in this context.

## **7.Acknowledgement**

The work reported in this paper was partly funded by the ESPRIT OMI/MODES project.

## **8.References**

- [1] BLENKHORN, P. and EVANS, D.G., A System for Enabling Blind People to Identify Landmarks: the Sound Buoy. IEEE Transactions on Rehabilitation Engineering, 5(3), 276-278, 1997.
- [2] HARPER, S., "Standardising Electronic Travel Aid Interaction for Visually Impaired People", MPhil Thesis, Department of Computation, UMIST, 1998.
- [3] HARPER, S., and GREEN, P.N., A Travel Flow and Mobility Framework for Visually Impaired Travellers, ICCHP 2000, Karlsruhe, Germany, 17-21 July 2000.
- [4] <http://www.bluetooth.com/>
- [5] <http://www.wapforum.org/>
- [6] MORRIS, D., EVANS, D.G, GREEN, P.N., and THEAKER, C.J., Object oriented Computer Systems Engineering, Springer-Verlag, 1996.