Limited Media Interface for AI Planning System

Alexander Nixon, John Levine, Austin Tate

Artificial Intelligence Applications Institute
The University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN, UK
me@alexnixon.co.uk, {j.levine, a.tate}@ed.ac.uk
http://www.aiai.ed.ac.uk/~oplan

Abstract

This paper is concerned with the implementation of a mobile telephone interface onto the existing AIAI hierarchical planner *O-Plan*, and the investigation into how to maximise the usability of this limited media interface, with minimum loss of content. The resulting *WOPlan* (Wireless O-Plan) interface was developed as a Java Servlet application which communicates with the O-Plan system and serves *WML* (Wireless Mark-up Language) pages to a WML-enabled mobile telephone. One interesting result of the WOPlan project was the development of a *plan execution* facility, whereby the user is presented with an ordered list of executable actions dependent on what actions have been executed so far. Another interesting result was the insight that the *reduction* of any graphical interface onto a limited medium may be a useful exercise in its own right, in that it necessitates the prioritisation of functional elements of that interface. We will discuss the design of the WOPlan system and assess its usability. Additionally we will give suggestions for improvements and for real-world applications.

1 Background

O-Plan(1)(5)[1] is an expansion-based hierarchical planner developed by the Artificial Intelligence Applications Institute at the University of Edinburgh. Planning problem domains (including possible tasks, constituent actions and relevant constraints) are defined in the *Task Formalism* (TF) language(10). The release version of O-Plan is written mainly in LISP with an X interface which offers the user the ability to specify problem domains, identify problem tasks and constraints, generate plans and view resulting plan details and hypothetical "world" states at various points in a plan. The same core O-Plan engine may also be accessed without using a graphical user interface (GUI) by exploiting the proprietary *TA Interface*, which consists of a defined set of LISP-style messages which may be passed to and received from a running instance of O-Plan to provide a subset of the functionality offered by the X interface.

[1] see `http://www.aiai.ed.ac.uk/~oplan/documents` for a full listing of O-Plan references

2 Aims and Motivation

The original aim of the project was to create a mobile, limited media interface onto the O-Plan system. What was envisaged was the case of the mobile human agent, equipped with a small, hand-held wireless device, attempting to access a planning server in order to request some kind of course of action dependent on that user's current situation. Available web-based demonstrations of O-Plan(7) propose problem domains involving various military disaster relief and evacuation operations, and it was thought that the mobile telephone or *Personal Digital Assistant* (PDA) could be a tool for plan delivery to mobile units in such a situation. Alternatively, a lone mobile user could access O-Plan to retrieve a plan to assist in a situation in which that user had insufficient experience. Someone who had no experience of engineering, for example, could retrieve a checklist to perform in the event of their car breaking down. The utility of such a system would depend not only on the design of the system itself, but also on the identification of a suitable problem domain, in particular a domain in which the users of the planner are likely to be on the move and in need of a course of action to solve an immediate task, and otherwise with no access to more conventional interfaces such as PCs. The name *WOPlan* (for "Wireless O-Plan") was given to the proposed system.

3 Complicating Factors

The design of a mobile interface onto O-Plan is made more difficult by the limited screen sizes of mobile devices, especially in the case of the mobile telephone. Development of interfaces onto O-Plan to date has allowed for the luxury of a full-sized terminal screen(4)(8). Issues of human-computer interaction which may be ignored (if not altogether forgotten) when using a full-sized terminal interface become unavoidable in the case of the limited media interface(18). Generally users of mobile devices ex-

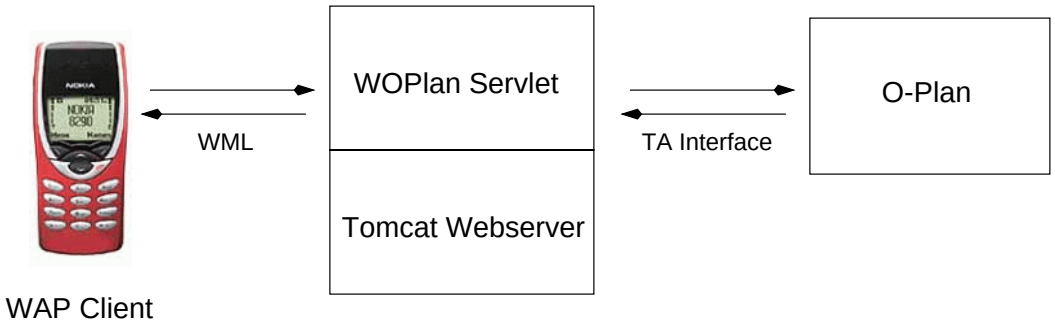

Figure 1: Architecture of *WOPlan*

pect their interaction with the device to be brief, whereas a user sitting down at a workstation is prepared for a more prolonged session. Browsing with a mobile device, especially with a mobile telephone, is (with current devices) slow and cumbersome; data entry is difficult and should be kept to a minimum. A mobile telephone system needs only be slightly poorly designed to be rendered unusable; this is especially true if the system is attempting to serve long lists of data (such as delivering a plan description), as long pages increase download times and make navigation even slower and more frustrating.

4 The Resulting System: *WOPlan*

WOPlan was developed as a web application which communicates with an instance of the core O-Plan engine and delivers *Wireless Mark-up Language* (WML)(12) to a connected client. The client may be any device with has a browser which conforms to the *Wireless Application Protocol* (WAP), such as a WAP-enabled mobile telephone. The WOPlan web application is a *Java Servlet*(20). In development and testing the *Nokia WAP Toolkit* WML browser emulator(22) was used in place of a physical WAP device, and the servlet was hosted within the *Tomcat Jakarta* webserver(21). The client initiates a session by connecting to WOPlan, which connects to the O-Plan server and initialises the service, and provides the user (on the WAP device) with a list of available problem domains. The user is prompted to choose a planning *domain* (defined as a *Task Formalism* file), then choose a *task* within that domain, and then to view, execute or evaluate the resulting *plan*, or to get a different plan that fulfils the same specified task.

4.1 WAP Client

This is the component with which the WOPlan user interacts. The user activates WOPlan by initiating an internet session on the WAP Client and navigating to the internet address of the second component, the WOPlan servlet. The WAP client could be any device with a WAP-enabled browser and internet connectivity, although WOPlan has only been tested in use with WAP emulator browsers running on a workstation.

The emulator used in development was the Nokia WAP Toolkit. Included at the end of this paper is a screenshot of the Nokia Toolkit emulator. Features to notice are the very limited screen size (only four lines of text), and the user interface objects. Directly below the screen are two arrow buttons (used for scrolling up and down a WML page), in between which is a single `Select` button (used for selecting whatever item is currently highlighted in the WML page). Below and left of the screen is the `Options` button, which when available and selected should display a context-sensitive list of options. Below and right of the screen is the `Back` button, which when available and selected should navigate the user back to the previous screen.

It is worth noting that this layout and combination of available keys is particular, though not necessarily unique, to this browser. Although all WAP-enabled browsers should fully implement all of the functional elements of WML as defined by the WAP Forum, the implementation is not constrained as to layout or number of buttons, etc. Generally, however, most mobile telephone WAP browsers to date have adopted a layout similar to this

Tool	*Desc*	*Location*
O-Plan 3.3	Hierarchical Planner	http://www.aiai.ed.ac.uk/~oplan
JAVA 2 SDK 1.2	Java Dev Kit	http://java.sun.com/products/jdk/1.2/
Jakarta Tomcat 3.1	Servlet Engine	http://jakarta.apache.org
WAP Toolkit 1.3	WAP Emulator	http://www.forum.nokia.com
RedHat 6.1	Linux OS	http://www.redhat.com
Emacs JDE	Java Env for Emacs	http://sunsite.auc.dk/jde

Table 1: Platforms/Applications used in the development of WOPlan

Nokia emulator. The Nokia WAP Toolkit, rather than any other emulator, was used because it is currently the most stable free browser emulator which is supported on a non-MS Windows platform, in that it operates as a Java archive, and so is entirely portable[2].

4.2 WOPlan Servlet

The servlet sits between the WAP Client and the core O-Plan system. It accepts WAP requests from the client (or from multiple clients simultaneously) and communicates with O-Plan, initially connecting to O-Plan as required and sending and receiving TA interface messages. All of the development work for the WOPlan system has focused on the WOPlan servlet; it is in this component that the logic specific to this implementation resides. The servlet dynamically creates WML pages, depending on the responses it is receiving from O-Plan, which are then sent to the WAP Client for browsing. These WML pages may themselves contain logic such as navigational directives or actions to perform after a certain length of time has passed. Although these directives are executed by the WAP Client, their source is the servlet.

4.3 O-Plan Server

O-Plan sits in the bottom tier of the architecture, responding to requests from the WOPlan servlet.

5 WAP/WML

5.1 WML

WML is based on *Extensible Mark-up Language* (XML) (13). It has been designed by the *WAP forum*(11) for delivering hypertext content to wireless devices, taking into considerations the limitations of such devices, including:

- Limited display size and user interface capabilities, including constrained data entry ability;
- Slow network connections;
- Limited system resources (memory and computational speed).

The complete WML specification defines a set of XML mark-up elements which together provide the following functional areas:

- Presentation and layout, including provision for text emphasis, simple table formatting, simple images (*Wireless Bitmaps*);
- Navigation, including hypertext links and the concept of *Cards* and *Decks*. Each page is presented as a card, but several cards may be grouped into a single deck for simultaneous delivery, thus saving network time;
- String parameterisation and state management. Variables may be used instead of explicit strings and their values are resolved by the WAP device at runtime. This may reduce network traffic.

5.2 Comparison of HTML and WML

5.2.1 Functionality

HTML is a visually-rich mark-up language appropriate for use with systems with large screens. WML allows a very restricted set of formatting commands, with no facility for plug-ins. Typically HTML browsers are very forgiving of sloppy mark-up encoding. For example, generally, unrecognised elements are ignored. WML, as an implementation of XML, requires more precision, and available WAP browsers are unforgiving of errors in WML. Tags must be properly nested with attributes correctly

[2] be aware that Nokia WAP Toolkit 2.0 has recently been released; it will run only on a Win32 platform (i.e. Windows 98/NT/2000)

specified. Generally, if a document violates the WML specification, it will cause an error and not be displayed. Additionally the variable and history mechanisms have to be taken into account in WML development.

5.2.2 Popularity

Recently, the usability of WAP devices and services has been the subject of widespread derision in the press(15); the disappointment in WAP services is not because of any failing inherent in the WAP or WML specifications, however, but rather primarily because:

- WAP is still young (1999) and WAP-compliant devices have only recently come on the market. The devices which are currently available have extremely limited screen sizes and no multimedia capabilities.
- WAP has been hyped by wireless industry leaders such as BT Cellnet as "putting the Internet in your pocket"(16). Invariably such hype encourages direct comparison between WAP services and more established web-based services delivered via HTML. With such functional limitations, WAP cannot possibly compete with HTML.

5.2.3 Purpose

WAP should not currently be regarded as a competitor to HTML, then, but rather a complementary internet technology. The flexibility of HTML allows for a greater range of applications available to the home or office based internet user, but the peculiar properties of the mobile WAP device provide potential for a range of useful applications for the user on the move.

6 Plan Viewing

A hierarchical plan consists of layers of *action nodes*, each deeper level representing an expansion of the nodes in the level above. Each node consists of two *enodes*, representing the beginning of the associated action, and its completion. Within each node, the `begin` and `end` enodes themselves each maintain a list of predecessor and successor enodes (of other nodes). These dependency lists fully specify the conditions for the legal execution of the plan. Each node may optionally contain some *time window* information. This is given as an earliest and latest time for beginning the associated action, and an earliest and latest time for completing it. The O-Plan GUI supports several

Figure 2: Example of *WOPlan* Narrative Display

modes of plan viewing. The most descriptive is the graphical representation, which denotes action enodes as boxes and dependencies between enodes as links. An example of this is shown in figure 5. Such a full representation of a plan would be unable to fit into the screen of the mobile telephone, however.

6.1 The Utility of a Narrative

Another mode of plan viewing which the O-Plan GUI supports is the creation of a plan narrative, a step-by-step ordering of the enodes which honours time constraints and enode dependencies. The narrative cannot represent the dependencies between enodes in the same way that the full plan description does (in that it is simply a one-dimensional ordered list of enodes), but rather attempts to show a typical (but usually not the only) execution trace through the actions of the plan. For a user who wishes to appreciate quickly the typical sequence of actions involved in executing a given plan, this mode of plan viewing could be more informative than the full graphical representation, and certainly more intuitive than a full list of node descriptions. In the case of the mobile telephone user, this narrative representation may be the only intuitive and economical way of presenting an entire plan on one page, and it is this representation which WOPlan implements.

6.2 The Narrative Creation Problem

The creation of the plan narrative constitutes a non-trivial problem. O-Plan solves this problem with a simple topological sort which attempts to take into account both enode dependencies and time windows. The procedure

begins by ordering the enodes by their time-windows, and then by name (so in the case that no nodes have any time window information, the enodes will simply be ordered by name). The topological sort then visits the enodes in reverse order. Each enode is visited by visiting all of that enode's dependent successors and then placing that enode at the front of the result. When all the enodes have been visited, the result is the plan narrative.

WOPlan approaches the creation of the plan narrative as a problem of sorted search rather than a topological sorting procedure. Although the topological search is more efficient than the sorted search, the reasons for using the search in WOPlan are:

- The sorted search procedure is more modular in design and allows for straightforward substitution of sorting criteria, which in turn can influence which enodes are presented earlier in the resulting plan narrative; the procedure is more flexible as a result;
- The sorted search is less trusting of the plan description than the topological search: if there is a problem with the plan description the sorted search will fail, whereas the topological search will complete incorrectly; as the creation of the plan description is not WOPlan's but O-Plan's responsibility, this cautious approach is more appropriate.

6.3 Example Narrative to Depths 3

The following narrative was created by the O-Plan GUI on the `house-1` problem domain, on the task `task_build_house`. Where both the begin enode and the end enode of a single node are consecutive in the narrative presentation, the node is called an `Action`. Where the begin and end enodes have been separated for some reason (as is the case when an upper level node has been expanded into its lower level nodes), each individual enode is shown with the appropriate qualifier `Begin` or `End`.

```
Plan execution starts.
0:00  Begin (build house).
      Action (excavate and pour footers).
      Action (pour concrete foundations).
      Action (erect frame and roof).
      Action (lay brickwork).
      Action (finish roofing and flashing).
      Action (fasten gutters and downspouts).
       Begin (install services).
      Action (install drains).
      Action (lay storm drains).
      Action (finish grading).
      Action (pour walks and landscape).
      Action (install rough plumbing).
      Action (install rough wiring).
       Begin (decorate).
      Action (pour basement floor).
      Action (install air conditioning).
      Action (fasten plaster and plaster board).
      Action (lay finished flooring).
      Action (install finished plumbing).
      Action (install kitchen equipment).
      Action (finish carpentry).
      Action (paint).
      Action (sand and varnish floors).
         End (decorate).
      Action (finish electrical work).
         End (install services).
         End (build house).
Plan execution finishes.
```

7 Plan Execution

7.1 Functionality

The final major piece of functionality which was added to WOPlan was the provision of a simple plan execution facility. This functionality is not available by default in the standard O-Plan GUI. In WOPlan, when a user chooses to `Execute` a plan, they are presented with a depth-ordered list of nodes which are currently executable, given what has been completed so far in the execution process. In between consecutive calls to the servlet, WOPlan recollects the current execution state of the plan. Initially the only node which is executable is **node-1**, which is always the `start` node. Accordingly, at the beginning of plan execution the user will see a list with only one action item in it, namely `start`. During execution, each of the items visible in the execution list is itself a link, which when selected displays a context-sensitive list of functions to perform on that node. Functions which may be performed on action nodes are:

- **Execute Node**. This function will be available if the node which has been selected is fully executable, i.e. the predecessors of the end enode of this node have already been executed during this session. Choosing this function will mark the node as complete.
- **Force Completion**. This function will be available if the node may be begun but not immediately ended

Figure 3: Example of *WOPlan* Execution Display

without some intermediate execution. This happens when a node is expandable, i.e. there are subactions which have to be completed before the entire node action may be completed. Choosing this function will mark this node and any subactions at any depth as complete.

- **Expand Node**. This is available when there are executable subactions at a level deeper than this node. Choosing this will cause these executable subactions to become visible in the plan execution facility.

The text of each node action in the execution screen is emphasised according to which of the above functions may be performed on it. Actions which may be immediately executed appear in unemphasised `small` text. Actions which may be completed by force appear in *`italic small`* text. Actions which are expandable appear in ***bold italic small*** text. When a function is performed on an action, the WAP client notifies WOPlan of the change, whereupon WOPlan updates its understanding of the execution status of any nodes which are affected by the change, and sends a refreshed list of executable nodes back to the client as a WML page. This cycle continues until either the user cancels the execution or the `finish` node is reached. If the finish node (always **node-2**) is executed, then the message `No More Action Items` appears, and so execution ends.

8 Assessment

8.1 Original Aims

This original aim has been achieved in that WOPlan provides reasonably stable, scalable and usable access to the O-Plan system through a mobile telephone. Although it does not provide all of the functionality which O-Plan, and in particular the TA interface, has to offer, it provides a useful subset of this functionality, and has addressed the core issues of plan review and execution through the narrative and execution facilities.

8.2 Usability

The following usability issues emerged from user testing.

8.2.1 Context-Sensitive Menu Confusing

WOPlan includes an `Options` menu which is modified depending on which menu items should be available to the user at any point during the use of the application. Interestingly, the users reported that the constantly-changing `Options` menu made them feel "uncomfortable", in that it made it difficult to assess, as a new user to WOPlan, exactly how much functionality there was included in the system, and they felt that new options were constantly making themselves available unannounced. They would have been happier if the same `Options` menu were available at all times, but that specific functional items on the menu were somehow disabled depending on what functionality should be available at that point. The tailoring of the options menu was intended to be a feature which increased usability. It may be that a more advanced user of the system would find context-sensitivity a helpful feature, but it is worth bearing in mind that novice users feel uncomfortable with an interface which is constantly shifting.

8.2.2 `Back` Key Required on All Screens

It was felt that a `Back` key would be a useful addition on all screens. Again, this is a problem which an advanced user would not encounter, as the `Options` menu always provides the ability to return to the previous command. Where this is not a possibility, a specific `Back` key is added. The exception to this is during plan execution, where once an execution is confirmed there is no way it can be cancelled. It is recommended programming practice(17)(18) for limited media interface design always to provide a `Back` key, and certainly it would be a useful addition to this application to add one to *all* pages.

8.2.3 No Indication of Plan Context

Unlike in the O-Plan GUI, there is no way within WOPlan for the user to recollect within which task, within which

problem domain, they are operating. This is a substantial omission, but could be easily addressed with a separate menu item for this kind of information.

8.2.4 Replan Functionality Confusing

The users were confused by the ability to *replan*. Technically what the `Replan` option allows is for the user to request a different plan for the same task. O-Plan searches for an alternative plan and, if one is found, returns it, otherwise the current plan remains. It is difficult for a user to assess a given plan without actually stepping through it, but even if the user decides, while viewing or evaluating the plan, that there is something unacceptable about that plan, there is no guarantee, if the user chooses then to replan, that the resulting new plan will not contain the same fault. Nor is there any way for the user to gauge how many possible alternative plans there are. One solution to this replanning issue would be to provide some facility whereby the user can dynamically specify a set of minimum acceptability constraints which O-Plan would then attempt to satisfy when searching for a plan. If the resulting plan is still unacceptable, the user should be able to modify the constraints before replanning. This kind of functionality exists on the current web-based demonstration of the core O-Plan system(9), but is not currently supported through the TA interface.

8.2.5 Add to Task Requirement

The users suggested that there should be some means of modifying or parameterising the existing tasks, as it would be unlikely that one of the published task definitions would be absolutely relevant to every user in every detail. The TA interface does include functionality to add constraints or actions to tasks, but there is no way to query O-Plan for information on what the available constraints or actions may be. Implementation of a robust `Add To Task` facility was therefore considered too ambitious within this project.

8.2.6 Plan Execution

Although the users were impressed with the simple and intuitive interface which the plan execution facility offered, they were critical of the inability to cancel and reactivate previously executed actions. A requirement of this kind of functionality would be some kind of list (perhaps in an incomplete narrative format) of actions completed so far during execution. This in itself would not be a difficult feature to implement. In addition it was felt that it would be useful to be able to *save* the execution state of a plan for an indefinite length of time, to allow the user to return to the execution at an unspecified later date to continue that execution. This kind of plan persistence would require the introduction of a database or flat file into the system's architecture, in order to store these execution states alongside the appropriate client details. It would also require some method of identifying clients beyond the lifespan of the servlet; if the webserver is shut down and restarted, the session details of the hosted servlets are lost. Perhaps one way of achieving this persistent identification could be through the use of simple logins, the details of which could be stored in a database or flat file.

8.2.7 Help Information

The provision of some kind of description of domains and tasks would be useful, as would the provision of more general *help* information on the system.

8.3 Performance

The WOPlan system has not been subjected to specific performance testing, although there appeared to be no performance problems during user testing, in which three separate clients maintained concurrent sessions with the WOPlan servlet. This testing was performed on workstations in the Department of Artificial Intelligence, which are Sun Microsystems' SPARC Ultra 5s with 360 MHz processors and 128MB RAM. During the test, the Tomcat webserver (running the WOPlan servlet) and O-Plan itself were located on separate workstations, as were each of the three Nokia WAP Toolkit clients.

8.4 Robustness

No catastrophic failures occurred during user testing, although exceptions have been noted during development testing. When identified these have been resolved. The servlet has been modified to serve a generic error message to the WAP client in the event of an exception, so at least if an exception occurs within the operation of the servlet, the failure is reasonably graceful, if not informative. The only possible action from such an error page is to `Restart`, which attempts a clean connect and initialisation with O-Plan, which would remove any residual problems. Recently the servlet has been modified to implement a *Single Thread Model*, along with session management(20), to avoid resource clashes when more than one client attempts to access the servlet at one time.

8.5 Scalability

WOPlan has modular, object-oriented design which should ease the process of modification or augmentation. All of the platforms used in the development of WOPlan are proven, enterprise-level technologies. Where possible, effort has been made to minimise the impact of running WOPlan. For example, the servlet retains a list of every single connection to O-Plan which is initiated on behalf of a WAP client. When the servlet is finally shut down (which normally only happens when the webserver itself is shut down), it iterates through this list, closing the connections if they still exist. An improvement on this would be to implement this closure for each client session whenever the session times out.

WOPlan has avoided *hard coding* configuration information such as the address and port of the O-Plan server, and the location of the directory which contains the TF source files. This information is *soft-coded* instead in a file, currently called `WOPlan.properties`, which resides in the same directory as the servlet. Changing these configurations should necessitate only editing this options file rather than recompiling the entire servlet.

9 Future Work

Evaluation of the usability of the WOPlan system (see section 8), identifies modifications which could improve the usability of the current system. In addition the following areas for future work have been identified.

9.1 Physical Testing

WOPlan has not yet been tested with a physical WAP device. In order to achieve this, a publicly-accessible webserver will need to be set up, hosting the WOPlan servlet. The security considerations associated with providing a public service will need to be addressed. A WAP-enabled wireless device will also be required. A full list of such devices is available at *AnywhereYouGo.com*(24).

9.2 Alternative Platforms

9.2.1 Client Platforms

It is likely that non-WML platforms and devices will become more widespread and popular in the near future. Subsequent development on WOPlan should be open to the possibility of a move to a non-WAP platform. Porting to a different client platform would require a significant, but local and contained, amount of work.

9.2.2 Server Platforms

Given that the WOPlan servlet is a Java application, and if the need or interest arises, it should be reasonably straightforward to port it to a Microsoft Windows NT/Windows 2000 server environment, running a webserver which can host servlets on Windows, such as Apache with JServ. The O-Plan server will still need to run on SunOS or Linux, although with some work that should be portable to Windows too.

9.3 Voice and Positioning

The investigation into the possible use of properties specific to the case of the mobile device was a secondary aim of the project. Two such properties which have been discussed are voice technology and mobile positioning technology. No provision for voice or location integration exists in currently available WAP devices, although they will certainly become available in the near future. One possibility for the former would involve the use of *VoiceXML*, an XML variant intended to "make Internet content and information accessible via voice and phone"(14). VoiceXML has the backing of industry giants IBM, AT&T and Motorola. The Motorola *Mobile ADK*(19) is a development environment for integrating VoiceXML and WML services. The provision of *Location Services* (LCS) as a standard for mobile devices is still currently at the design stage. It is likely that some kind of service based on *Global Positioning System* (GPS) technology will be available to GSM (*Global System for Mobile Communications* - the current European standard) telephones in the near future.

9.4 Suitable Problem Domains

Beyond the integration of mobile-specific technologies, the WOPlan system could certainly benefit from the engineering of a Task Formalism problem domain which is likely to have more relevance to a mobile user than the currently available O-Plan TF domains. The issue of the creation of a useful and relevant TF file is inextricably connected to the issue of providing the facility within WOPlan to adapt plans according to changing circumstances, and to specify minimum acceptability constraints, as discussed in section 8.2.4. If a truly usable problem domain is conceived, it must be written in such

a way as to allow parameterisation of acceptability conditions, and to provide, as available choices, the kind of tasks and constraints which are likely to arise in the case of that mobile user.

9.5 Plan Execution Research

The execution facility provided with this version of WOPlan is little more than a prototype, but it offers interesting possibilities for further development and research. Firstly it could certainly be improved, augmented and made more usable within the context of the WOPlan system. Perhaps more importantly, however, its simple, ordered, one-dimensional format, with action items emphasised according to what may done with those items, could provide a basic template for any system with a mobile limited media interface which is attempting to deliver courses of action (COAs) to human agents on the move.

9.6 Two Applications

By way of a summary for future work, we provide here two examples of what we see as potential real-world applications of a mobile interface into a planning system.

9.6.1 Multi-User COA Delivery

This system would be a multi-user, multi-role planner with at least two separate user interfaces: an office-based planner with a full screen interface would be used to coordinate multiple agents and update any courses of action according to changing circumstances, whereas a mobile interface would be used by mobile agents to send status updates to the central planning unit and receive and execute the courses of action. To build this kind of functionality would require detailed research into user roles, and integration of logic to handle the marshalling of multiple users. Although such a system would fit comfortably into the physical, three-tier architecture of WOPlan, it would involve such complex re-engineering that the logical design of the system would be almost unrecognisable from what is currently there.

9.6.2 Web-based Journey Planner

An intra-city journey planner could be implemented with relatively little alteration of the existing WOPlan system, except for addressing the usability issues specified at the beginning of this chapter, and designing an appropriate TF problem domain. An example of its use would involve a mobile user accessing the system from a WAP-enabled telephone to request a plan to travel from their current location to some destination within the same city. The planner would have access to information about all the bus, tram, train and underground stops in the city, as well as information about the walking time between each of these stops. With some careful thought this information could be encoded in a Task Formalism problem domain which would include the relative costs of using each form of transport and the costs of walking between specific stops, etc. The planner would generate a course of action to transport the user from their current location to the specified destination by the shortest route possible within certain acceptability constraints. This course of action could then be delivered back to the user as a narrative or a plan execution. The application could also provide the facility to reject certain stages of the journey (for example, if the user knew that traffic was bad in a particular area, or wished to avoid a certain station), which would force the planner to replan, avoiding those particular stages. In due course, with the advent of location services, the user need not even enter their initial location, as it would be automatically queried by the system.

10 Conclusions

One of the unexpected conclusions to result from this project is that the implementation of an existing system on a limited media device may be a useful exercise in its own right, in that it forces the designer both to appreciate what are the essential informative features of the system and to consider the ways in which the usability of the system can be maximised.

An interesting part of the work on WOPlan was the development of the plan execution facility. Although the current facility is little more than a prototype, there is clearly a good deal of work that could be done, and progress made, in the area of delivering executable courses-of-action to mobile human agents. This is a clear case of value added in the case of the mobile user, not through the use of voice or positioning or any other kind of technology, but purely for the reason that the user is mobile and is actually in a position to carry out actions, unlike the office-bound user (for most problem domains).

We are confident that the WOPlan system provides a valuable tool on which to base further research into the area of mobile planning and plan delivery. The adaptable design and use of freely-available, robust, scalable plat-

forms should allow straightforward modification of and addition to the existing code if desired.

Figure 4: Screenshot of *Nokia WAP Toolkit* Browser Emulator

Figure 5: Graphical Representation of Plan for Task `build.house` in Domain `house-1`, Showing Typical Node Connectivity

References

[1] Tate, A. and Currie, K. **O-Plan: the Open Planning Architecture**
Artificial Intelligence vol 52, 1991
`http://www.aiai.ed.ac.uk/~oplan/documents/1991/91-aij-oplan.ps`

[2] Tate, A., Drabble, B. and Dalton, J. **The Use of Condition Types to Restrict Search in an AI Planner**
Proceedings of Twelfth National Conference on AI(AAAI-94), Seattle, July 1994
`http://www.aiai.ed.ac.uk/~oplan/documents/1994/94-aaai-typed-conditions.ps`

[3] Tate, A., Drabble, B, and Kirby, R. **O-Plan2: An Architecture for Command, Planning and Control**
Intelligent Scheduling, 1994
`http://www.aiai.ed.ac.uk/~oplan/documents/1994/94-is-oplan2.ps`

[4] Tate, A. and Drabble, B. **O-Plan's PlanWorld Viewers**
Proceedings of the 14th UK Special Interest Group on Planning and Scheduling, Essex University, November 1995
`http://www.aiai.ed.ac.uk/~oplan/documents/1995/95-sigplan14-viewers.ps`

[5] Tate, A., Drabble, B. and Dalton, J. **O-Plan: a Knowledge-Based Planner and its Application to Logistics**
Advanced Planning Technology, May 1996
`http://www.aiai.ed.ac.uk/~oplan/documents/1996/96-arpi-oplan-and-logistics.ps`

[6] Drabble, B., Dalton, J. and Tate, A. **Repairing Plans on the Fly**
Proceedings of the NASA Workshop on Planning and Scheduling for Space, Oxnard CA, USA, October 1997
`http://www.aiai.ed.ac.uk/~oplan/documents/1997/97-jpl-repair.ps`

[7] Tate, A., Dalton, J. and Levine, J. **Generation of Multiple Qualitatively Different Plan Options**
Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems (AIPS-98), Pittsburgh PA, USA, June 1998
`http://www.aiai.ed.ac.uk/~oplan/documents/1998/98-aips-plan-options.ps`

[8] Tate, A., Levine, J., Dalton, J. and Aitken, S. **O-P^3: Supporting the Planning Process using Open Planning Process Panels**
Proceedings of the AAAI-99 Workshop on Agent Based Systems in the Business Context, Orlando, USA, July 1999

[9] Artificial Intelligence Applications Institute **O-Plan version 3.3**
30th May 2000
`http://www.aiai.ed.ac.uk/~oplan/release/index.html`

[10] Tate, A. and Drabble, B. **Task Formalism Manual**
July 1995
`http://www.aiai.ed.ac.uk/~oplan/documents/ANY/oplan-tf-manual.ps`

[11] Wireless Application Protocol Forum, Ltd **WAP Forum**
`http://www.wapforum.org`

[12] Wireless Application Protocol Forum, Ltd **Wireless Application Protocol Wireless Markup Language Specification Version 1.3**
`http://www1.wapforum.org/tech/documents/WAP-191-WML-20000219-a.pdf`

[13] T. Bray et al **Extensible Mark-up Language (XML), W3C Proposed Recommendation 10-February-1998, REC-xml-19980210**
February 10, 1998
http://www.w3.org/TR/REC-xml

[14] VoiceXML Forum **VoiceXML Forum**
http://www.voicexml.org/forum_1.html

[15] Schofield, J. **Shall we scrap Wap?**
The Guardian, Thursday 31st August, 2000
http://www.guardianunlimited.co.uk/Archive/Article/0,4273,4056871,00.html

[16] BT Cellnet **WAP's it all about?**
BT Cellnet press release 3rd April, 2000
http://www.btcellnet.co.uk/SiteGen4/Feature/181850/1/

[17] Nokia Corporation **Nokia WAP Toolkit Version 2.0 Developer's Guide**
Nokia Forum, June 2000
http://www.forum.nokia.com

[18] Palm, Inc **Palm OS Programmer's Companion**
Palm Computing Platform Development Zone, 2000
http://www.palmos.com/dev/tech/docs/palmos/CompanionTOC.html

[19] Motorola, Inc **Motorola Mobile Internet Exchange**
Motorola Products and Services 2000
http://www.motorola.com/spin/mix/faqs.html

[20] Sun Microsystems, Inc **JAVA™ Servlet Technology**
http://jsp2.java.sun.com/products/servlet/

[21] The Apache Software Foundation **The Jakarta Project: Tomcat**
http://jakarta.apache.org/tomcat/

[22] Nokia Corporation **Nokia WAP Developer Forum**
http://www.forum.nokia.com/wapforum

[23] AnywhereYouGo.com **AnywhereYouGo.com**
Miscellaneous resources for wireless developers
http://www.anywhereyougo.com/ayg/ayg/Content.po?name=waptools/Emulators

[24] AnywhereYouGo.com **List of Wireless Devices**
http://www.AnywhereYouGo.com/ayg/ayg/wap/devices/Index.po

[25] Isys Information Architects, Inc **Interface Hall of Shame**
http://www.iarchitect.com/mshame.htm