Supporting Goal-Based Interaction with Dynamic Intelligent Environments

Thomas Heider and **Thomas Kirste**[1]

Abstract. Modern technical infrastructures and appliances provide a multitude of opportunities for simplifying and streamlining the everyday life. However, many of the systems available today – such as the typical feature-loaded audio and video components – are not always efficiently usable for the average person. But, in an environment where features abound, the easy access of these features more and more becomes the key quality criterion for the user.

We present a planner-based approach to helping the user to interact with such complex infrastructures. Specifically, we concentrate on the application domain of networked infotainment systems and home control. Also, we describe the architectural concept, which makes it possible to integrate classical Artificial Intelligence technology – such as planning and scheduling – into the domain of networked consumer appliances within the scope of a multimodal assistance system.

The work we present is part of the EMBASSI-project, a joint project with 19 partners from industry and academia that aims at establishing an interoperable system infrastructure for multimodal and multimedia assistance systems.

1 Overview

Planners are software components that allow the automatic creation of strategies for reaching a given goal based on a given set of possible actions. There is a substantial body on planning tools available in the literature [2, 12, 3]. Also, there are numerous applications – from logistics [11] to robotics [5] and spacecraft control [3].

However, in the area of consumer applications, plan-based approaches are not yet very prominent.

Which, on the other hand, is rather curious: it is a well known fact that todays consumer electronics – specifically in the infotainment sector – provide more functionality than the average user is willing to master.

Providing such a system component with planning capability – this is, relieving the user of coming up with a control strategy by himself (which would require him to know all possible operations of his infrastructure) – might allow to make more of the available functionality *accessible* for the user.

In this paper, we present such a planner-based approach to helping the user to interact with complex technical infrastructures of the everyday life.

It is important to note that we do not claim to present any fundamentally new approaches to planning itself. The point of our paper is rather to outline how existing planning technology can be utilized for interacting with networked consumer appliances, and how it can be embedded into dynamic multimodal component infrastructures.

2 The application domain

A human being's daily activities – professional or private – are based on a broad range of interactions with numerous external objects: controlling the TV at home, driving a car, buying a ticket from a vending machine, visiting an exhibition, discussing project plans with colleagues, setting up a multimedia presentation in the conference room, editing documents, delegating travel planning to a secretary, and so on. These objects make up the user's personal environment.

As computers are becoming more and more ubiquitous, moving from the desktop into the infrastructure of our everyday life, they begin to influence the way we interact with this environment – the (physical) entities that we operate upon in order to achieve our daily goals. The most important aspect of future human-computer interaction therefore is the way, computers support us in efficiently managing our personal environment. This might be called the *ecological* level[2] of user-interface design.

At the ecological level, we look at future developments from the perspective of helping a user in achieving his individual goals and purposes by providing computer-based assistance. The vision is to have the computer acting as a *mediator* between the user and his personal environment.

In addition, in order to minimize the cognitive (and sensomotorical) gap between human/computer interaction on the one side and human/environment interaction on the other side, *natural* (anthropomorphic) *interaction* should be supported through multimodal user interfaces, which integrate *e.g.*, classical GUI, speech interaction and gesture-based interaction

The prime objective of the EMBASSI project is to provide such a goal-based, natural interaction with the typical technical infrastructures of our everyday life. EMBASSI [6] is a focus project supported by the German Ministry of Education and Research (Bundesministerium für Bildung und Forschung, BMBF) within the strategic research area Man-Technology-Interaction. With 19 partners from industry and academia and a time scope of four years, EMBASSI intends to provide an integrated approach to the development of assistants for our everyday technologies.

The primary application area for EMBASSI are technical infrastructures of the non-professional everyday life – in particular, application scenarios are being developed in the home, automotive, and public terminals environments.

[1] Fraunhofer Institut for Computer Graphics, Rostock, email:{kirste,theider}rostock.igd.fhg.de

[2] "Ecology is the scientific study of interactions of organisms with one another and with the physical and chemical environment." So this notion quite well captures the essence of the above discussion.

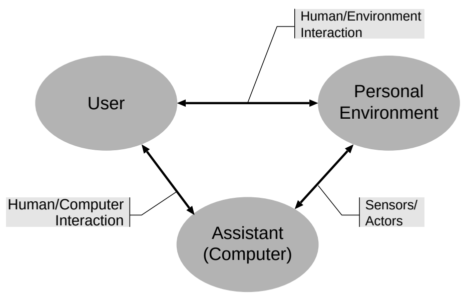

Figure 1. The ecological interface

EMBASSI is conceptually based on two important paradigm shifts:

- Transition from essentially unimodal, menu-based dialogue structures (with a fixed interaction vocabulary provided by the system) to polymodal, conversational dialogue structures (with an unrestricted interaction vocabulary provided by the user).
- Transition from a function-oriented interaction with devices to a goal-oriented interaction with systems.

While these paradigm shifts are being discussed in the research community for some time now, it is a substantial challenge to make these results accessible to the user of, *e.g.*, home entertainment infrastructures. This is the goal of EMBASSI.

The paper is further structured as follows: Section 3 gives an overview of the challenges of supporting goal-based interaction with dynamic systems. In Section 4, we outline the architectural framework of an overall system infrastructure, into which planning and scheduling has to be intgrated. In Section 5, we look at the resulting requirements for a planning component and outline our current approach. Finally, in Section 6 we give an outlook over the next steps for integrating planning into home environments.

3 Goal-based interaction with dynamic systems

When looking at networked consumer appliances, it may not be immediately obvious why a control concept based on planning technologies should be considered. However, there are two important aspects that justify an investigation of this approach:

3.1 Function-based vs. goal-based interaction

When interacting with technical infrastructures of the non-professional private life – the notion "devices" will be used from now on – we are educated to think of interaction in terms of the individual "functions" these devices provide: functions such as "on", "off", "play", "record", etc.. Different devices have different functions, similar functions in different devices behave differently, and staying on top of all features is not altogether easy. When interacting with devices, we select, parameterize, and then execute functions these devices provide. When these functions are executed, they cause an effect: a broadcast is recorded on videotape, the light is turned brighter, and so on.

Of course, it is the effect we are interested in when using a device, not the functions we need to select in order to get the desired effect.

This is the basic idea of goal-based interaction. Rather than requiring the user to invent a sequence of actions that will produce a desired effect ("goal") based on the given devices and their capabilities, we allow the user to specify just the goal ("I want to see 'Chinatown' now!") and have the system fill in the sequence of actions leading to this goal (Find the media source containing media event "Chinatown". Turn on the TV set. Turn on the media player – *e.g.*, a VCR. Position the media source to the start of the media event. Make sure the air condition is set to a comfortable temperature. Find out the ambient noise level and set the volume to a suitable level. Set ambient light to a suitable brightness. Set the TV input channel to VCR. Start the rendering of the media event.).

Once we abstract from the individual devices and their functions, we arrive at a system-oriented view, where the user perceives the *complete* set of (networked) devices as a single system of interoperating components that helps him in reaching his specific goals. Once we abstract from (device) actions and have the system communicate with the user in terms of (user) goals, we also have effectively changed the domain of discourse from the system's view of the world to the user's view of the world.

In order to support this kind of interaction, we need a system that is able to reason about the effects a device operation has with respect to the environment state as perceived by the user.

3.2 Dynamic configuration

Dynamic configuration, the ad hoc extension of an EMBASSI system by new devices and components, is another important architectural goal of EMBASSI.

Imagine, a new device – *e.g.*, a light source – is plugged into an EMBASSI system and the user wants to see Chinatown again. How should the system handle the new device when trying to set the ambient light level to a comfortable state? Of course, the system components that are responsible for producing solution strategies for the "Chinatown" goal need to know that the new device provides functions that change the environment variable "ambient light"!

This means, in order to solve the problem of dynamic system extension by new devices, without simply fixing the set of allowable functions and environment variables, we need to find a way for making the state-changing semantics of device functions explicitly visible to the system. As in the previous section, we must enable the system to reason about the effects available actions will have on the environment.

Note that a similar consideration holds when looking at the interpretation of user utterances and user goals. Imagine adding a device to the home entertainment infrastructure that supports a whole new set of user goals – such as adding printer, which makes it possible to create hardcopies of video stills. Here, the concept of a "hardcopy" is completely new to the system, and the dialogue managers need to be told what kinds of user sentences do refer to this concept and what kinds of goals they do represent. Here too we need a mechanism for reasoning about goals.

4 The EMBASSI architecture

After defining the central conceptual components of the EMBASSI-framework, the next step is to define a reference architecture that allows an *implementation* of such a system.

A central requirement for an EMBASSI architecture is that it should support technical infrastructures that are built from individual components in an ad hoc fashion by the end user. This situation is for instance common in the area of home entertainment infrastructures, where users liberally mix components from different vendors. Also, it is not possible to assume a central controller – any component must be able to operate stand-alone.

Therefore, such an architecture should meet the following objectives:

- Ensure independence of components,
- Allow dynamic extensibility by new components,
- Avoid central components (single point of failures, bottlenecks),
- Support a distributed implementation,
- Allow flexible re-use of components,
- Enable exchangeability of components.

Furthermore, the architecture should clearly identify the essential conceptual protocol ontologies that are relevant within the system.

4.1 General concepts

The generic architecture that we have developed within EMBASSI (Figure 2) is a pipeline approach to the problem of mapping user utterances to environment changes. Each "level" in the architecture represents one function within this pipeline, while the level interfaces have been introduced at "meaningful" places, separating different ontologies. These "ontologies" (the sets of objects that are discussed at a level) become visible at the level interfaces. The level interfaces make up the EMBASSI protocol suite.

Each level consists of a number of processes ("components") that co-operatively implement the level's function. Processes can be added or removed dynamically: suitable co-ordination mechanisms at each level are responsible for managing the interactions between the processes at this level. There is deliberately *no* central co-ordination component.

The use of this rather fine-grained level-model in conjunction with the feature of dynamically adding or removing processes at each level allows us to create systems that can be incrementally built and extended in an *ad hoc* fashion, using modular components. Specifically, it allows us to build interoperable systems, where different components are provided by different vendors and where components are added and removed over time by the end-user.

Also, this allows us to collect components in a "technology toolkit", from which specific assistant systems can be built by simply "plugging" these components together.

An important aspect of the generic EMBASSI architecture is the context manager component. It is responsible for managing the system's view of the world – information about the user, resource profiles, the environment, and also the availability and state of the individual EMBASSI components. Attached to the context manager, we have sensors to obtain biometrics and environmental information.

4.2 The MMI levels

An EMBASSI system has to accept multimodal utterances which it needs to translate into goals before it can begin to think about changing the environment. According to the architecture in Figure 2, this translation process can be broken down into three distinct steps:

1. First we translate physical interactions into atomic interaction events (lexical level).
   The transformation of physical user interactions into unimodal atomic events is done by the I components (I = Input).
2. The stream of atomic interaction events is then sent via the $Event$ interface to the F components (F = Filter). These components are responsible for inter- and intra-modal aggregation of atomic events into amodal *sentences* (syntactical level).
3. The stream of sentences arrives at the D components (D = Dialogue manager). D components are responsible for translating sentences into goals denoted by these sentences (semantical level). Also, D is responsible for managing inter-sentence relations (dialogue memory) and dialogue dynamics (such as turn-taking).

The process is reversed when producing output: D sends amodal output "sentences" to the R components (R = Renderer) which in turn map these sentences to multiple atomic output events for the available output channels, the O components.

4.3 The assistance levels

The assistance levels operate on goals which are identified by the MMI levels. The process of mapping goals to changes of the environment consists of the following steps:

1. A components (A = Assistant) take goals (which specify state changes of the environment) and try to develop strategies for fulfilling these goals (strategy level). These strategies are called *plans*.
   There is no predefined way to produce a plan. Some A components may use hardwired plans, others could use decision trees or even complete inference systems. We will look at a planner based approach in more detail in the following section.
2. The plans are sent to the X components (X = eXecution control), which are responsible for the (distributed) scheduling and execution of the plans. (This means, the EMBASSI-architecture advocates a two step planning policy, as described *e.g.* in [10], where strategy planning (A-level) and execution scheduling (X-level) are distinct processes.)
   The scheduling-components ensure the correct sequential processing of the individual steps in a plan. Also, they are responsible for managing the parallel execution of multiple plans and for the management of execution resources needed by the plan.
3. Finally, individual action requests are sent to the (abstract) devices (device control level), the G components (G = Gerät – German for "device"). The G components execute the action request by employing the physical device they control, thus changing the state of the environment and causing an effect as intended by the user.

5 An in-depth look at planning

After looking at the overall structure of the EMBASSI-architecture, we will now look at the requirements that have to be considered when trying to integrate automatic planning functionality into such an environment.

5.1 The planning problem

The planning component (a component at the A-level) receives the goal identified by the MMI-level. It has the job to find a strategy that changes the environment from its current state to the goal state. This can be understood as a classical planning problem:

- The goal is given as a set of positive and negative literals in the propositional calculus.
- The initial state of the world (resp. the state of the system and the environment-condition which is known to the system) is also expressed as a set of literals. As defined in the architecture, the Context Manager supplies these initial state values – all devices (G components) post their actual state to the Context Manager.

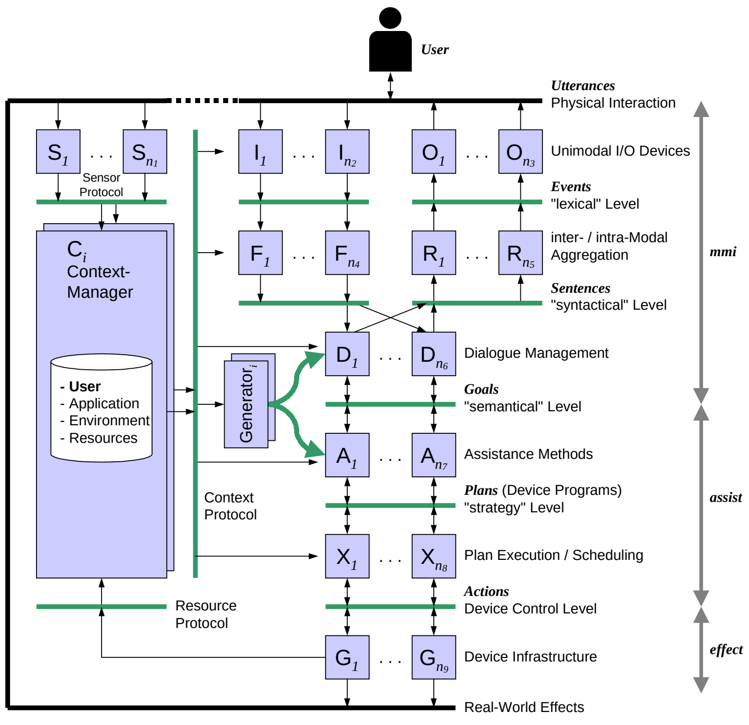

Figure 2. Generic EMBASSI architecture

- The actions provided by the available devices ("operators") have to be characterized using a suitable definition language. As outlined in Section 3, it describes the action's relation to the environment: it contains a set of preconditions that must be true before the action can be executed and a set of changes, or effects, that the action will have on the world. Both the preconditions and effects can be positive or negative literals. The modelling of the device functions as planning operators is most important: it is the base for providing goal-based interaction with a dynamic infrastucture.

The critical aspect here is the expressive power of the model used for describing device operators, which needs to be strong enough to capture at least the operational semantics of todays consumer appliances.

5.2 The planning domain model

Providing a suitably expressive operator definition language is not a completely trivial requirement when looking at the host of features included in modern infotainment systems. But with expressiveness comes computational intractability – the more expressive a language is, the more computation is required to reason about sentences in that language. On the other hand, the solution capability of the planning system determines the space of the possible functionality of the device components. For example, the choice of discrete operators obviously excludes devices that provide continuous functions.

So finding the right balance between expressiveness and computational tractability is very important for our application domain. Specifically, we have to consider the following aspects:

- Computational resources in a consumer environment are inherently limited, but the system needs to provide a snappy response to the user.
- Operator sets for devices must be compact in order to keep the number of operators that have to be managed by the system small and in order to simplify the creation of (compatible) operator sets by device vendors.

Our experience from the modelling of our domain has shown that we need a planning environment that supports conditional effects and disjunction in the preconditions – this allows a compact representation of device operator sets.

Furthermore it is mandatory to have universal quantification in the preconditions and the effects. This for instance allows to define operators that apply to an arbitrary number of objects – which is extremely important in an environment that is dynamically extensible. (As example, consider the operator `lights-out` given in Figure 3, which turns off all light sources.)

Also, it is an advantage, if the planning system supports domain axioms, because they provide a convenient way to decouple operators from the environment: Instead of describing the environmental preconditions and effects of an operator in the operator's definition itself, we only describe the operator in terms of the device's internal state. Environmental aspects are attached by providing suitable domain axioms (as done, *e.g.*, by the axiom `shutter-dark` in Figure 3, which derives the effects of a venetian blind on the room lighting from the blind's internal state). This approach simplifies an incremental definition and extension of the environment state – which relieves us from the complex task of coming up with a complete environment model before defining the first operator . . . This advantage has to be contrasted with the fact that some of the fastest planning systems available today – *e.g.* FF[7] – unfortunately do not support domain axioms.

ADL [8] and PDDL [1] are examples for languages providing the properties listed here, where PDDL seems to become the standard.

```
(define (axiom shutter-dark)
 :context (and (shutter ?x)
               (or (time_night)
                   (not (open ?x))))
 :implies (dark ?x))
(define (operator lights-out)
 :precondition (forall (lightable-object ?x)
                       (dark ?x))
 :effect (darkness))
```

Figure 3. Sample for an operator with universal quantification and an axiom with disjunctive precondition

Finally, the planning system should generate partially ordered plans (rather than totally ordered plans), so that independent actions can be executed in parallel. Moreover, it is thus possible that the scheduler can apply refined strategies – such as least cost scheduling – for determining the concrete execution sequence.

Based on these considerations, we have chosen the UCPOP planner [12] for the first running prototype of our planning assistant (the system is still under development). UCPOP uses a formalism that handles a subset of ADL and is the predecessor of PDDL. Although it is an older system, it offers all the possibilities that our present application domain requires, and it has been available in a stable version for our system infrastructure at the time we had to fix our platform choice.

5.3 Joint environment ontologies

As we mentioned before, the description of the component functions and capabilities as operators for the planning domain is essential. In order to support the *interoperability* of devices provided by different vendors, we need a shared understanding of the common environment domain they operate upon – a uniform ontology (besides, of course, agreeing on a common operator description language). Standardized environment ontology concepts such as "brightness", "temperature", ... make it possible to develop the components' operator definitions independently from each other. Different vendors have to adhere to these ontology concepts as an explicit specification of the environment aspects for their specific planning subdomains. If different components use common concepts for the same features, *e.g.* "brightness" for the capability of a lamp and of a venetian blind (by daylight), a cooperation is feasible.

The vendor of a component characterizes its products in accordance with the specification of the ontology and the potentialities of the chosen problem-specification language. The planning operators will reasonably abstract from the device's concrete internal state and use a simplified state model that is tailored towards attaching the operators' environmental effects.

6 Summary

With this paper, we have presented a system infrastructure supporting goal based interaction with dynamic environments, which is being developed as part of the EMBASSI project. The goal of the project is the development of a framework for an intuitive, multimodal, goal based interaction with technical infrastructures of everyday life. We have presented our backbone architecture of EMBASSI, our current concepts for supporting goal-based interaction within this architecture, and the resulting requirements for the integration of a planning system into the EMBASSI-infrastructure.

EMBASSI system prototypes have been built for the application areas of consumer electronics & home control, car infotainment, and point-of-sales / point-of-information terminal access. A dynamic planning system is currently integrated into the home application scenario; the complete system has been on display at different conferences and fairs.

Current work on the planning component focuses on the following aspects:

- Development of a well-structured planning ontology for the home scenario, based on the experiences made in the first project phase. (This includes the operator definitions as well as the structure of the environment state.)
- Extension of the planning component by functions for plan repair, execution monitoring and conditional planning.
- Investigation of domain design support tools such as TIM (Type Inference Module) [4] or DKEL (Domain Knowledge Exchange Language) [9] for supporting the definition of structured planning domains.

7 Acknowledgements

The work underlying this project has been funded by the German Ministry of Education and Research under the grant signature 01 IL 904 G 0. The content of this publication is under sole responsibility of the authors.

REFERENCES

[1] AIPS-98 Planning Competition Committee. PDDL – The Planning Domain Definition Language. Tech Report CVC TR-98-003/DCS TR-1165, Yale Center for Computanional Vision and Control, October 1998.

[2] Blum, A., Furst, M. Fast Planning Through Planning Graph Analysis. *Artificial Intelligence*, 90:281–300, 1997.

[3] Chien, S. et al. ASPEN - Automated Planning and Scheduling for Space Mission Operations. Technical report, California Institute of Technology, Jet Propulsion Laboratory, June 2000. `http://planning.jpl.nasa.gov`.

[4] Fox, M., Long, D. The Automatic Inference of State Invariants in TIM. *Journal on Artificial Intelligence Research*, 9:367–421, 1998.

[5] Giralt, G., Alami, R., Chatila, R., Freedman, P. Remote operated autonomous robots. In *Proc. International Symposium Intelligent Robotics*, Bangalore, India, 1991. International Society for Optical Engineering (SPIE).

[6] Herfet, T., Kirste, T., Schnaider, M. EMBASSI: multimodal assistance for infotainment and service infrastructures. *Computers & Graphics*, 25(4):581–592, 2001.

[7] Hoffmann, J., Nebel, B. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal on Artificial Intelligence Research*, 14:253–302, 2001.

[8] Pednault, E. ADL and the State-Transition Model of Action. *J. Logic Computat.*, 4(5):467–512, 1994.

[9] Scholz, U., Haslum, P. Decoupling Domain Analysis and Planning. `http://www.intellektik.informatik.tu-darmstadt.de/~scholz/Planning/Papers/dkel_unpublished.ps.gz`, 2001.

[10] Smith, D., Frank, J., Jonsson, A. Bridging the Gap Between Planning and Scheduling. *Knowledge Engineering Review*, 15(1), 2000.

[11] Tate, A., Drabble, B., Dalton, J. O-Plan: A Knowledge-Based Planner and its Application to Logistics. In *Advanced Planning Technology*. Morgan Kaufmann, 1996.

[12] Weld, D., Penberthy, J. UCPOP: A Sound, Complete, Partial Order Planner for ADL. In *1992 International Conference on Principles of Knowledge Representation and Reasoning*, pages 103–114, 1992.