UIML-DISL Q/A

General Issues

Q: Where can we find general information on DISL?

Most information can be found at my website where I listed my publications:
www.c-lab.de/vis/robbie/publications.html

Information on DISL is in following papers:

- Interactive Multimodal User Interfaces for Mobile Devices
- Multimodal Dialog Description for Mobile Devices
- A Novel Dialog Model for the Design of Multimodal User Interfaces (short paper)

One paper submitted for publication but can be sent by private mail. This is basically a detailed version of the short paper and deals more with the underlying model.

Then we have a master thesis about DISL but unfortunately in German language, but I can extract some important things like the DTD and send it again by private mail.

Q: Where is DISL being used now and how?

At the moment, DISL is still evaluated and used mainly to control applications such as the WinAmp-player. So you could compare this application domain with the Pebbles project from CMU. At our University we foresee usage mainly in multimodal control of ambient computing environments and we have set up a small lab as an experimental home environment where we could control several aspects of the home and entertainment functionality with different devices and modalities e.g. speech, gestures etc. also depending on context.

Design Issues

Q: Why DISL does not encompass the peers section as in UIML? Should DISL therefore be considered as a subset of UIML?

Yes. We relied on external transcoding as we found some limitations in peers (mainly due to linear mapping). For DISL we mainly concentrated on the dialog model, however allowing contact to backend application. Generic widgets are not proper UIML though and should be put in vocabulary.

Q: Why do you think the "for-each" of DISL is a cleaner solution than borrowing the XSLT "for-each" element? With little adaption I think it could accomplish the same. In Mozilla XUL I can imagine javascript being used as the solution for this problem (though I am not sure).

I do not really see any significant differences between different for-each implementations or semantics that basically perform the same task. For-each was not part of our original DISL specification (and we did not implement it yet), so I thought it might be a good addition to cope with the "calculator problem". I was not aware that XSLT-style had been proposed already.

Formal Issues

Q: How expressive is DISL (e.g. Turing-complete)? Since it is based on DSN and your definition of DSN ("DSN basically defines transitions from different state/event combinations to new states") seems a little bit like a petri-net, is DISL comparable to petri-nets for describing system behavior?

It is Turing Complete. Yesterday we set together and proved that we can map every Turing Machine Programme on a DISL description.

Well, it is not really a Petri-net approach. We do not have a concept of tokens wandering around and firing transitions. We have to investigate yet, if we are able to map our transitions to Petri-nets. However since high-level Petri-nets are Turing-complete as well, there should be a way.

Q: Could you elaborate on "UIs typically have a more complex structure that can be modelled as a cross product over the state space." Is this statement always true?

Not always, see your calculator example, or voice driven interfaces with linear structure. However, complex UIs as for word processors handle multiple states which should be looked after e.g. use of menus / buttons / preferences to modify fonts, paragraphs, colour etc.

Q: I think it (UIML + DISL) fits nicely in the model-based approaches we were discussing during the last meetings. DISL could be appropriate to represent the "dialog model". Do you agree on this?

Exactly, that is where we planned it to be. In fact pure DSN and extensions like DSN/2, ODSN have been used in the past for dialog modelling at UPB. The other parts of DISL are a UIML subset and the "Generic Widgets" could also be in a specific UIML vocabulary.

Multimodality

Q: DISL is relying on so called 'generic widgets'. In this sense, how can a generic widget be independent of the interaction modality? For me, a generic widget is toolkit independent, but not modality independent since graphical modality is assumed.

That is a general misunderstanding I perceived with many people when talking about DISL, as "widget" indeed is too much used in conjunction with graphical UIs. The Idea was to simplify UI elements that they can be used with almost any modality, for example making them able to be rendered in linear dialogs (voice) or in two dimensional spaces (GUI). Maybe the best way to communicate this idea is to drop the term "widget" and talk about generic interaction- and presentation elements.

Q: DISL is said to be supportive for multimodal dialogs. How do you cope with true multimodality, that is several modalities simultaneously, and not only one modality at a time?

A better way would be to say is that DISL is prepared for multimodality and the underlying implementation fetches interactions from different inputs but processes them sequentially. So we can for example indeed use different input devices to control an application, but this is dealt by processing the interactions sequentially as they arrive. So multimodality in DISL could be compared to multitasking on single processor systems.

What is missing in DISL yet, are features to control synchronization of the modalities, e.g. which input/output devices may or must work in parallel to provide meaningful interactions.

Q: do you support the CARE properties for multimodal UIs and if so, how?

We did not have the concept of a component model in mind but more the well known concept of "one model and many interfaces", which can also mean that we might have several DISL specifications for each device / modality instead of using different languages. However we support transformations to other languages, but then the modelling power of DISL will be lost.
For the individual CARE properties it is like this:

- Assignment: Implicitly (e.g. mapping speech to DISL events)

- Equivalence: Also Implicitly. It does not matter if we use gesture or speech command to control the same application DISL treats input the same and converts to calls to backend application.
- Complimentary: Not yet supported -> Synchronization is missing

Q: Do you support combinations of modalities as defined in the Tycoon framework and if so, how?

I was not aware of this Framework before, thanks for pointing me to it. It is the same as for the previous answers: Synchronization is needed.

Q: do you intend to transform DISL to other languages such as InkML, MRML, etc.?

In principle we are open for it, but couldn't commit to it due to lack of time and resources.
However we experimented with transformations of UIML to HTML, WML and VoiceXML in the past and might pick up these activities for DISL. I assume it is challenging to flatten the behavioural part though (mapping multi state transitions to single state transitions). Also intended is to transform higher level models to DISL for example in the scope of the Nomadic Media Project.

Implementation Issues

Q: Do you plan to implement DISL for uiml.net? ;-)

I wish I had time:
a) To learn .net
b) to implement DISL traits in uiml.net

Q: Have any renderers been built for DISL and if so, to what languages?

At the moment we have a J2ME renderer which needs the serialized DISL, and works on several mobile phones for example Siemens SX1. Planned are transcoders to e.g. HTML, XForms, VoiceXML, WML but in that case the expressiveness of the dialog model may be lost.

Standardisation Issues

Q: What are the lessons learned through designing DISL that can make UIML a better specification?

General lessons:

- UIML as Meta Language may be sometimes complicated to handle
    - shortcuts preferred
    - our design not compatible but shorter because of fixed vocabulary
- Some misunderstandings regarding UIML. Maybe specification should contain more detailed examples

Specific lessons (Design Decisions for DISL):

1. Lack of UIML implementation for mobile phones
    - client/server architecture needed
    - minimize calls through the network
        - synchronization with backend application only needed when commands are passed or application state is requested, otherwise autonomous state transitions
        - need for advanced dialog model
    - renderer and UIML documents have to cope with sparse resources
        - serialized format
2. UIML peers mapping constraints (linear mapping)

- need for external transformations (peers not needed)
- calls to backend can be placed in behavior (peers not needed)

--> This would be major change in UIML and can be avoided

3. Multimodal Interaction
   - Avoid target specific mapping --> needed generic vocabulary

Q: How do you see DISL and the final UIML standard interacting/complimenting each other down the road?

I don't see problems in DISL complementing to UIML as we used an UIML subset mainly to limit implementation efforts and the extensions of this subset might very well be beneficial to UIML.

In the short term (for the current standard), UIML could profit from DISL's dialog model. I see the "generic widgets" as a possibility for a set of vocabulatories that could be standardized as add-ons.

For our further research I see still improvements in DISL like specifying real multimodality / synchronization, object-oriented features from ODSN etc. To this end we would further develop DISL as the basic language is less complex (faster development), which could then again flow into future UIML standards (assuming that there might be a future UIML 4.0 standard).