ScIML: Model-based Design of Voice User Interfaces

Jörn Kreutel
University of Potsdam,
Applied Computational Linguistics Lab
kreutel@ling.uni-potsdam.de

Abstract

We will introduce ScIML, a domain specific language for voice user interface (VUI) creation that is based on the generic expressive means of the Unified Modelling Language. In particular, we employ UML statecharts for interaction flow modelling.

1 Introduction

In the course of the last decade and beyond, significant research has been carried out in the field of dialogue management, in general, and spoken dialogue systems, in particular (see, e.g., (Traum, 1996; Seneff et al., 1998; Larsson et al., 1999; Bohus and Rudnicky, 2003). On the other hand, available approaches and technology for developing commercial dialogue systems – which we will term as *Voice User Interfaces* (VUIs) throughout this paper – have so far not advanced beyond the simple form-filling mechanism underlying VoiceXML (Oshry, 2004). The latter, at the same time, exhibits a severe lack of modularisation as far as a separation of concerns between dialogue management, on the one hand, and prompt and grammar creation, on the other, is concerned.

Missing transformation achievements between research and the 'voice industry' may partially be due to the fact that the latter strongly requires a *visual* representation format for VUIs that makes transparent the functionality of a VUI to all stakeholders in a project, be it technical or business staff. In addition, industry projects require that any aspect of a VUI, in particular its interaction flow, be principally subject to particular *design* decisions, i.e. it needs to be *hand-craftable*. Both requirements, however, are outside the primary scope of research on dialogue management. In fact, the concept of a generic dialogue management component which implements a range of domain independent interaction routines may even be seen as marking a contrary position, as long as its functionality is not foreseen to be at least overridable by domain specific implementations.

Given these findings, this paper will outline the basic ideas underlying the *Scene based Interaction Modelling Language* (ScIML),[1] which approaches the issue of voice user interface creation from the perspective of *model based user interface design*. The expressive means of ScIML are a range of VUI-specific concepts that are formalised as extensions of the UML meta model (Group, 2004), whose visualisations are well established within the IT industry. In particular, ScIML employs UML statecharts (Harel, 1987) for dialogue management purposes, i.e. it relies on a generic formalism for describing the behaviour of complex event-driven systems.

Methodologically, ScIML adheres to an account of user interface modelling that is known as OO&HCI (*Object Oriented Modelling and Human Computer Interaction*). It conceives of UI design as involving a range of different interrelated modelling activities. ScIML adopts this approach and supports the respective activities through appropriate expressive means which exploit the current state of the art in dialogue systems research. In particular it employs the two basic concepts of *dialogue acts* (Poesio and Traum, 1998; Bunt and Girard, 2005) and of

[1] The ScIML notion of *scene* is based on proposals for GUI modelling in (de Paula, 2002).

Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue, pages 235–238,
Antwerp, September 2007.

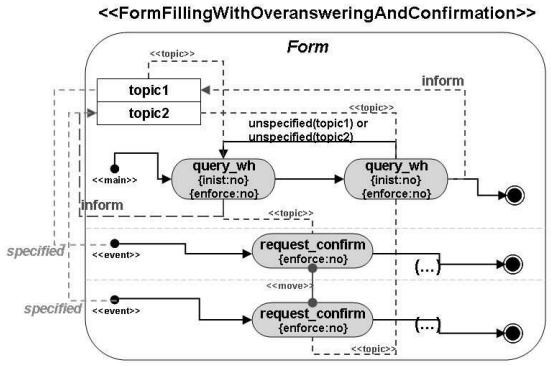

Figure 1: Integrated representation format for ScIML, depicting, main and event workflows, topicality relations, non-local response spaces and move formation

grounding (Clark and Brennan, 1991; Matheson et al., 2000).

2 ScIML Modelling Activities and Artefacts

ScIML VUI Referent Models provide, for each activity of an underlying **Task Model**[2], a structured description of the entities that are addressed by the VUI and/or the user in the course of the realisation of that activity. Referent models, hence, specify the potential *topics* of conversation for some VUI. The ScIML meta model assumes three abstract referent types: *activity*, *entity* and *event*, which are concretised as *domain activity* and *VUI activity*, *domain entity* and *VUI entity* and *domain event* and *VUI event*, respectively. Examples for the latter are, e.g., failures of speech recognition or missing inputs on the part of the user.

Over the set of referents of a ScIML referent model, we further assume the relations of *occurrence* between events and activities, and the one of *involvement* that specifies associations between entities, on the one hand, and activities or events, on the other, as well as between two activities. We further assume that entities may be *complex*, which is reflected by a *constituency* relation between entity referents. Activities and events, on their part, will be considered as *complex referents* by nature. Methodologically, these relations serve as a starting point for referent identification on the basis of a task model. Assuming that each activity of the latter corresponds to an activity referent in the referent model, it is possible to determine both the entities that are involved in some activity and the events that may occur in it.

For authoring a VUI referent model, we use UML class diagrams whose classes and associations are *profiled* on the basis of the assumed referent types and relations between them.

ScIML Interaction Structure Models define the *topical structure* of interactions over some given VUI referent model. For this purpose, an interaction structure model identifies, first of all, a set of *scenes* which can be conceived of as topically coherent contexts that span over sequences of *moves* by the user and the VUI. *Moves*, on their part, are modelled as sets of *dialogue acts*.[3] Both for scenes and for dialogue acts, the referents that serve as their respective topics are provided by the constituents of the VUI referent model. For *scenes*, it is additionally required that their topics be *complex* referents. Interaction structure models, further, describe which *domain functions* for accessing backend data and executing transactions are required for each scene's realisation.

Interaction structure models are authored as class diagrams that define the *topicality* association between the members of the referent model and the assumed *scenes* and *dialogue acts*. They further specify the association between dialogue acts and those scenes in whose realisations the acts are involved. Note that interaction structure models are *structural* models in the sense that they merely define the set of scenes for some VUI application, as well as, for each scene, the set of dialogue acts that may be involved in its realisation. Both the actual dialogue flow by

[2] ScIML Task Models describe, at a coarse-grained level, the *activities* that are actually or supposedly – from a user's perspective – supported by the VUI. Activities may be either *domain activities* or *VUI activities*, and are structured in the sense that some activity may involve the realisation of a range of subactivities.

[3] ScIML's notion of *dialogue acts* is based on the idea that for the purpose of VUI modelling, dialogue acts can be described by a generic *dialogue act type* and a domain specific referent that identifies the *topic* of the act. This proposal withdraws from thinking of the *propositional content* of dialogue acts as rich semantic representations of system and user utterances. Instead, it assumes that the *illocutionary force* of dialogue acts – or their *update effect*, in other terms, see e.g. (Poesio and Traum, 1998) – operates on assignments of referent values.

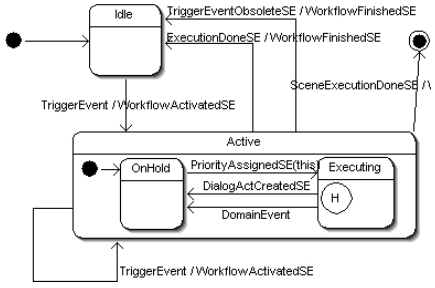

Figure 2: Generic Processing Model for a Workflow in ScIML

means of which a scene is realised and the clustering of elementary dialogue acts into *moves* is outside the scope of this model type.

ScIML interaction flow models define, for each scene of an interaction structure model, the realisation of this scene through a set of *event-triggered workflows*. A workflow is a sequence of state transitions between *VUI dialogue acts*, *domain functions* and *sub scenes*. In this approach, dialogue acts performed by the user are conceived of as a particular type of *event* that may trigger the initialisation of a workflow or a state transition within some active workflow.

The concepts underlying this type of model are described, in more detail, in the following section.

ScIML presentation models comprise, on the one hand, a *VUI move model* that defines *patterns* of VUI dialogue acts that constrain the move formation on the part of the VUI. On the other hand, they include the definition of a *response space model*. The latter assigns, for each occurrence of a VUI dialogue act in the interaction flow model, a set of user moves that are assembled, on their part, from dialogue acts. Presentation models are authored within an *integrated representation format* for ScIML that is exemplified by figure 1.

3 Statecharts-based interaction flow Modelling

Given our notion of *scene* as the domain with regard to which interaction flow will be specified, the purpose of an interaction flow model is to determine, for each realisation of a scene, whether it is in one of the following *activity states*:

- the performance of a dialogue act
- the performance of some domain function for backend data access or transaction execution
- the realisation of some subscene

For dialogue act states, ScIML assumes that the corresponding dialogue act will only be realised if its preconditions hold and as long as they *do* hold. This way, e.g., a *form filling algorithm* can be reconstructed – as in figure 1 – by a sequence of query_wh dialogue acts which will only be realised if their respective referents have not been specified before. Thus, ScIML's notion of dialogue act states is able to account for dialogue flow phenomena like Overanswering.

However, rather than specifying the control algorithm for a scene as a single FSM, we propose to think of it as being described by a *set* of *workflows* that define transitions over the above states and that are *triggered by events* of the following types:

- *VUI Events*, which indicate a failure to recognise a user's input, the missing of input by the user, or any other exceptional behaviour particular to the usage of a VUI.
- *Dialogue Act Events*, which signal the performance of a dialogue act by the user.
- *Grounding Events*, which express a change of the grounding status of some referent. Grounding events will be caused, on their part, by the performance of dialogue acts.[4]
- *Domain Events*, which may be thrown during the execution of some domain function. The call of a domain function may, on its part, be triggered by the occurrence of a grounding event, e.g. a specified event with regard to some referent or set of referents.

For each scene there will, further, be one *main workflow* that will be triggered upon entering the scene and that describes, e.g., a *form-filling* flow.

[4] We assume that a referent's grounding status may be either unspecified, specified, i-grounded, c-grounded, i-rejected or c-rejected. The notions of i-grounded and i-rejected, on the one hand, and c-grounded and c-rejected, on the other, reflect the two dimensions of *reliability* with regard to a user's intention and *validity* with regard to the given application domain.

Note, however, that triggering of a workflow may not immediately result in executing it. Instead, there is, additionally, a generic *workflow prioritisation* algorithm that determines the ordering in which workflows will be executed. E.g., in case a trigger event for some workflow occurs outside the context of the corresponding scene, this workflow will, further, generically be prioritised over the scene's main workflow. Particularly in voice portals that offer a variety of services under single entrance point, these processing routines allow that a user may not only identify a desired service, but may also provide more information with regard to the latter's referents.

As a complementary process to workflow triggering, SciML allows to discard workflows that have been made obsolete by events that occurred after they have been initialised. With regard to *workflow obsoletion*, SciML assumes that obsoletion conditions can be derived from the triggering conditions of a workflow.[5]

For authoring, SciML employs an abbreviation of the actual statechart representation. Using statecharts in their particular version as UML *activity diagrams*, only the particular flow inside the generic workflow execution model – i.e. the content of the Executing state – will be explicitly authored. As figure 1 shows, these workflows will be specified in the *parallel* regions of a scene state.

4 Outlook

The SciML execution model described in the previous section has been verified on the basis of the existing Apache reference implementation of an SCXML interpreter. SCXML is an XML syntax for statecharts and has recently been proposed by the W3C (Auburn et al., 2005) as a standard for UI interaction flow control. It is also meant to enhance VoiceXML towards the creation of more flexible 'advanced' VUIs. However, statecharts lack expressive means for this particular purpose. SciML, in contrast, shows how statecharts can be intuitively *profiled* for VUI modelling and thus means to contribute to the uptake of SCXML in the voice industry.[4]

[5] For example, if an i-rejected grounding event occurs for any referent value involved in a workflow trigger, or if a specified event specifies an alternative value for the latter, the affected workflows will be obsolete. This will not be the case, however, if the workflow contains a subscene state that, on its part, specifies a workflow for the respective event.

References

RJ Auburn, Jim Barnett, Michael Bodell, and T.V. Raman. 2005. State Chart XML (SCXML) scate machine notation for control abstraction. Working draft, W3C.

Dan Bohus and Alex Rudnicky. 2003. Ravenclaw. In *Proceedings of the Eighth European Conference on Speech Communication and Technology (Eurospeech 2003), Geneva, Switzerland*.

Harry Bunt and Yann Girard. 2005. Designing and open, multidimensional dialogue act taxonomy. In *Proceedings of Dialor 2005, the 9th Workshop on the Semantics and Pragmatics of Dialogue*. LORIA, Nancy/France, June 2005.

Herbert H. Clark and Susan E. Brennan. 1991. Grounding in communication. In L. B. Resnick, J. Levine, and S.D. Teasley, editors, *Pespectives on Socially Shared Cognition*. APA.

Maira Greco de Paula. 2002. Projeto da interação humano-computador baseado em modelos fundamentados na engenharia semiótica: Construção de um modelo de interação. Msc thesis, Pontificia Universidade Catolica do Rio de Janeiro.

Object Management Group. 2004. UML 2 Meta Model. Specification, OMG.

David Harel. 1987. Statecharts: A visual approach to complex systems. *Science of Computer Programming*, 8:231–274.

Staffan Larsson, Peter Bohlin, Johan Bos, and David Traum. 1999. Trindikit 1.0 manual. TRINDI Deliverable 2.2, University of Göteborg.

Colin Matheson, Massimo Poesio, and David Traum. 2000. Modelling grounding and discourse obligations using update rules. In *NAACL*.

Matt Oshry. 2004. Voice Extensible Markup Language (VoiceXML) 2.1. Recommendation 2.1, W3C consortium.

Massimo Poesio and David Traum. 1998. Towards an axiomatisation of dialogue acts. In *Twente Workshop on Language Technology*.

Stephanie Seneff, Ed Hurley, Raymond Lau, Christine Pao, Philipp Schmid, and Victor Zue. 1998. Galaxy-ii: A reference architecture for conversational system development. In *Proceedings of ICSLP 98*.

David Traum. 1996. Conversational agency: The trains-93 dialogue manager.