Multimodal Dialogue Management in the COMIC Project

Roberta Catizone, Andrea Setzer, Yorick Wilks
Department of Computer Science
University of Sheffield
{R.Catizone, A.Setzer, Y.Wilks}@dcs.shef.ac.uk

Abstract

The next generation internet applications will feature not only the ability to understand spoken and written natural language text, (pen) gestures and body postures, they will also and importantly be able to engage with the user in a natural dialogue about the application. In this paper we will describe the design of a multimodal dialogue and action management module, part of the COMIC demonstrator, which is aimed at these next generation applications. The design uses well understood structures like stacks and augmented transition networks in a novel way to obtain the flexibility needed for mixed-initiative dialogue. We also show how this is applied to the application of the COMIC demonstrator - bathroom design.

1 Introduction

Even though the usage of the Internet has grown over the last years, the usage of internet services such as eCommerce and eWork did not meet the expectations. One of the main factors is the steep learning curve that users have to tackle before they can effectively use the service. The basic functionality, such as browsing through articles, adding items to a shopping cart etc. are familiar to the user, and thus the difficulty must lie in non-intuitive interfaces.

Next generation applications aim to overcome these immense difficulties by using 'perceptual user interfaces' which combine human-like sensing and perceiving capabilities with social skills and conventions. This requires a rich multimodal communication environment, in which natural language can be freely used, or only with few constraints. With perceptual interfaces, the user is always given the opportunity to interact in the way that is convenient to him or her.

To support a multimodal communication environment, the system has to be able to understand speech and natural language typed text, gestures, pen input, facial expressions, and body posture. One of the main features is the ability to engage in a natural dialogue with regards to the application with the user.

The COMIC project, addresses the problems and has the objectives of (a) developing software that will improve the usability of eCommerce and eWork services and (b) demonstrating the usability in form of a novel application for the support of bathroom design.

In this paper we will concentrate on the dialogue and action management (DAM) module, which is the module responsible for maintaining a natural dialogue with the user. We will describe our approach as well as justify the reasons for designing it the way we did in section 3. The basic design of the DAM involves a stack, containing Augmented Transition Networks (ATNs) and a control mechanism for pushing and popping the ATNs. Section 4 will talk about how ATNs are created. The dialogue relies also on an applica-

tion oriented ontology, which will be described in section 4.1. The penultimate section of the paper, section 4.2, addresses the issue of how particular ATNs are chosen to deal with the task at hand.

2 The COMIC Project

COMIC (COnversational Multimodal Interaction with Computers) is a recently launched IST project[1] which will last for three years. The main aim of COMIC is to define generic cognitive models for multimodal interaction and to evaluate these in a number of demonstrators.

The main demonstrator COMIC will build will demonstrate the significantly increased usability that is the result of adding multimodality to conventional point and click applications. It is based on an elaborate bathroom design tool, which was developed by the COMIC partner ViSoft[2] and with which bathrooms can be designed and decorated. The demonstrator, which adds multimodality to the tool, allows a naive user to design and decorate his or her own bathroom. Using the modalities of speech, pen gestures, and facial output, the system assists the user in entering the measurements and outline of the bathroom and choosing the decoration and sanitary ware for the bathroom. The demonstrator will be implemented in both English and German, and will integrate separate modules developed at the partner sites. A simplified overview of the architecture, showing these modules, can be seen in figure 1.

Automatic speech (ASR) and pen recognition (APR) comprise the input processing module. ASR focuses on the recognition of paralinguistic information, whereas APR addresses handwriting recognition, 2D and 2.5D pen gestures.

Fusion, the responsibility of DFKI, analyses and interprets multiple concurrent input modes simultaneously to create a comprehensive representation of the communicative goals and actions of the user. The meaning representations from this processing stage are hypotheses for reasoning in the dialogue and action module (DAM) in the context of the ongoing human-computer interaction.

The dialogue and action Management, being developed at the University of Sheffield, is the focus of this paper and will be described in detail in the following sections.

A Fission module then receives meaning representations from the DAM which represent the goals and actions of the system which can be user or system-driven. Fission decides the actual lexical content and which output channels to use. Three main output channels are available: verbal channels, including speech and text; head channels, including facial expressions, gaze, lip movements, and whole-head gestures such as nodding; and other visual channels, including drawings and graphics.

The output processing module consist of three sub-modules reflecting these output channels. Speech Generation realises the verbal channel, the avatar is the vehicle for the head, and the ViSoft

3 Designing the Dialogue and Action Management Module in COMIC

3.1 Initial Considerations

Any survey of this field right now might suggest that we may be in something of the same position as the field of Information Extraction (IE) when Jerry Hobbs(Hobbs, 1993) wrote his brief note on a generic IE systems, on the assumption that all functioning IE systems contained roughly the same modules doing the same tasks, and that the claimed differences were largely matters of advertising, plus dubious claims to special theories with trademarked names. However, we may be in a worse position with dialogue systems because, unlike IE, there is little or no benchmarked performance with which to decide which modules and theoretical features aid robust dialogue performance. The lack of an established evaluation methodology is, by common consent, one of the main features holding back robust dialogue development methodology.

We cannot even appeal to some accepted progression of systems towards an agreed level of maturity, as one can in some areas on NLP: even very primitive dialogue systems from long ago contain features which many would associate with very sophisticated systems: Carbonell's POLITICS (Carbonell, 1979) seems to be just a series of questions and answers to a complex knowledge base,

[1] Contract number IST-2001-32311

[2] www.visoft.de

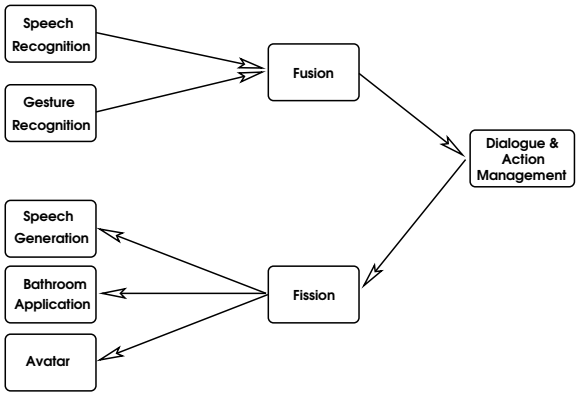

Figure 1: Simplified COMIC architecture

and might therefore be deemed a very simple dialogue grammar, lacking even a global structure of greetings and goodbyes. But it is clear that he considers the system to deploy coded forms of goals, beliefs and plans, which one might take as a sufficient property for a more developed class of systems.

Again, PARRY (Colby, 1971) the most developed and robust of the early dialogue systems, might also seem to be simply a dialogue grammar at heart, yet it very clearly had the goal of informing the user of certain things and, even though it had no explicit representation of goals and beliefs, it did have a primitive but explicit model of the user.

We have assumed in COMIC that a plausible DAM system must be able to have at least the following functionalities:

(a) determine the form of response locally, where appropriate, to dialogue turn pairs, where appropriately means in both pragmatic (i.e. dialogue act functional) and semantic terms (i.e. give correct answers to questions if known).

(b) have some form of representation of a whole dialogue, which means not only opening and closing it appropriately, but knowing when a topic has been exhausted, and also how to return to it, if necessary, even though it is exhausted from the system's point of view. This functionality need not imply an explicit global representation or 'grammar' of a dialogue.

(c) have appropriate access to a data base if there is to be question answering on the basis of stored (usually application relevant) knowledge.

(d) have appropriate access to a database that can be populated if information is to be elicited from the user as part of a basic task.

(e) have some form of reasoning, goal/intention representation, user modelling and planning sufficient to perform these tasks, though this need not imply any particular explicit form of representation or mechanism for implementing these functionalities.

(f) have some general and accessible notion of where in the performance it is at any moment: this can take the form of where in the dialogue it is or where in the overall task performance (if there is a task) or both. The only exception to this would be Loebner type systems, whose only function is to keep talking coherently and not repeat themselves.

Not all the above are essential, since some depend on the overall application/functional environment: e.g. (c) above is needed only for systems that know something and can answer questions about it, and (d) above applies only to tasks that can be described as 'form filling'. The COMIC application, and most conceivable ones, requires both of those. A trickier matter will be (f) where it is by no means agreed what applications require explicit models of users and of their goals, beliefs and intentions etc. Again, the power of any reasoning system implied by (f) is by no means clear, given both the certainty, from Church, that inconsistencies cannot in general be detected, as well as the overall mediocre performance of general AI reasoning systems.

Our proposal in this most complex and inviting area will be to so design the DAM so that min-

imal systems can serve some of these functions initially, though without preventing the substitution of richer ones later on. At every stage, our prejudice will be in favour of limited, constrained, knowledge-based systems, rather than general inference mechanisms of unconstrained power.

The functionalities above will probably not give rise to much dispute, nor will the sorts of module in a DAM to implement them, at least not if the Hobbs-vanilla assumption at the beginning is broadly right. However, something must be said about other desiderata and constraints (before proceeding to a DAM sketch), ones that follow both from our own theoretical prejudices and from the global design of COMIC within which we now have to work.

The global design of COMIC implies (unsurprisingly) what one might term an (almost) fully interlingual DAM: one in which the DAM deals only with coded representations of content and function as input and output (i.e. "first order English" more or less) and not with surface language strings (in German in this case). We write "almost" because we expect the COMIC DAM to get the German input string (though we do not want to attempt a second parsing of it) but not the German output string, which is the job of Fission to create (or its other-modal equivalent). However, this conventional (indeed archaic) modularisation of the DAM leads to two well-known problems.

(g) (We will continue to label key issues alphabetically, even though they are not all of the same type) There is the issue of a DAM not knowing the output string, and the problem that E.M.Forster expressed as "How do I know what I think till I see what I say". DAM does not know what COMIC says, so how can it know what it thinks?

In a sense this is absurd, at least in terms of AI/NLP orthodoxy, but there are other serious problems attendant on no-string-access that we will later call the "information retrieval problem": how do we find the relevant task/response structure without the surface information in input and potential output? There is also the psychological problem for researchers, known since the 1960s and immortalised in McDermott's paper (McDermott, 1981), that NLP researchers cannot work with, or even create, structures they cannot understand, and structures in first order English may or may not be comprehensible to the DAM team, particularly if made up by another (Fusion team) working in another natural language. Let's call this just a minor development problem, known to everyone outside the purely statistical/connectionist NLP camp (if that still has any members).

The DAM, in our view is thus a Chinese Room, in Searle's sense (REF), receiving and emitting coded forms it may not understand, and that may include the researchers as well as the algorithm itself. The solution is obvious and we shall just announce it as a principle:

(h) The DAM team will decorate its own structures and rules with strings of arbitrary length consisting of English and German words, for both retrieval and mnemonic functions. These may in many cases correspond to the most plausible generated output in English or German or both and these will be passed on to Fission who can do with them what they like. It should be obvious that retrieval of relevant action frames is more likely with a set of retrieval terms/wider than the internal predicate set. An important research issue is how Fusion, DAM and Fission teams in a highly modularised dialogue system are to understand recursive expressions in the first-order content language in the same way, as developers and as consumers of those codings. One could compare here the little discussed issue in machine translation as to how analysis and generation teams in an interlingual MT project know they understand the intermediate representations in the same way.

Another assumption follows naturally here and, if accepted, will have considerable consequences for Fission:

(i) the format and range of slots to be filled in the interface objects that pass from a DAM to a Fission module should be identical to those that pass from Fusion to DAM. In some sense this is obvious, and anyone who dislikes it should produce convincing cases where it is inadequate, which will mean finding information types beyond the standard Fusion-to-DAM information set of (roughly): semantic content, dialogue act, (possibly) intention form, language string, modality. It may well be inappropriate to pass the latter from

DAM-to-Fission except as a record of the original input (and unless that is independently unavailable to Fission from the dialogue history itself).

None of this prolegomenon reaches the core of the DAM design, but just (re)states assumptions, many of them obvious. We can now add some further ones, not normally discussed in the literature much but well-known to those who have had to produce working systems, particularly those subject to evaluation or assessment, as we have.

The key problems for dialogue system performance, and therefore reasons for failure, are:

(j) the inability of a dialogue system to find the relevant structure/frame that encapsulates what is known to the system about the subject under discussion, normally one introduced by user initiative. This is the main form of what we referred to earlier as the information retrieval (IR) problem in dialogue management and, unsurprisingly, we shall use IR methods to solve it, by using, as one main determinant of what structure to load as the current structure, overlap of terms between input and (our own indexing of potential) output. The same method will be asked to provide constant confirmation that the current structure is the appropriate one, over and above considerations from heuristics about expected input and global/local dialogue act sequencing.

This last can be thought of as principles of not-getting-lost or knowing-where-we-are, yet it is not a problem that can be solved by any single consideration or simply assigned to any magic module. Another problem that may be no more than (j) under another description is that of knowing-what-to-do-now:

(k) this is a problem central to all dialogue systems, and quite different strategies are out in the field: e.g. the Rochester-style strategy (Allen and Perrault, 1980) of the system taking a definite, and possibly wrong, line with the user, relying on robust measures for revision and recovery if wrong, as opposed to a hesitant (and potentially irritating) system that seeks constant confirmation from the user before deciding on any action. We shall also opt for the former strategy, and hope for sufficiently robust recovery, while building in implicit confirmations wherever appropriate.

But the issue here is narrower than simply one of having additional modules: it requires a core dialogue engine that is both a simple and perspicuous virtual machine (and not a lot of data/links and functionalities under no clear control) and which can capture (given good data structures) the right compromise between push (user initiative) and pull (system initiative) that any robust system must have. Our COMIC DAM sketch below, now being implemented, is intended above all to capture this combination of perspicuity (for understanding the system and allowing data structures to be written for it) and compromise.

3.2 Choosing a Level of Structure

The opening remark above about vanilla-DAMs was not strictly true in one respect: there still is no consensus over whether a DAM should be expressed as a set of rules (from finite state to context sensitive) or as some forms of script/frame or network that expresses a set of rules, rather in the way a syntactic RTN expressed a set of context free rules. The same opposition was present in AI planning theory between rule-driven planners and systems like SRI's STRIPS that pioneered more structural objects consisting of expected default actions. The TRINDI system (Cooper et al., 1999) is expressed as basically a set of rules (plus other structures, such as QUD, Questions Under Discussion) whereas the WITAS system (Lemon et al., 2001) was initially, at least, based on ATN-like structures.

The argument between them is, at bottom, is about (i) how much stereotopy one expects in a dialogue and (b) how much is it worth collecting all rules relevant to a subtopic together, within some structure or partition? Stereotopy in dialogue is closely connected to the notion of system-initiative or top-down control, which is strongest in "form-filling" systems and weakest in chatbots. If there is little stereotopy in dialogue turn ordering then any structure, like an ATN, risks being over-repetitious, since all possibilities must be present in many nodes. If one must always be ready to change topic at any turn, it can be argued, then what is the purpose of being in a higher level structure that one may have to leave; the answer being that it is possible to be always ready to change topic but to continue on if change is not

forced. As with all frame-like structures since the beginning of AI, they express no more than defaults or preferences. To have QUDs is no more or less than to express this preference in a different way.

In the COMIC DAM we shall opt for an ATN system with a single stack (with one slight modification) and argue that the WITAS argument for abandoning ATNs–namely, that structure was lost when a net is popped–is easily overcome, and that their alternative (dialogue trees) brings new problems of its own. We envisage ATNs of radically different sizes and types: complex ones for large scale information eliciting tasks, and small ones for dialogue control functions such as seeking to reinstate a topic.

Our argument will be that the simplicity and perspicuity of this (well understood and easily written and programmed) virtual machine has benefits that outweigh any disadvantages, and in particular the ability to leave and return to a topic in a natural and straightforward way. As we shall see below, this is a complex issue, and the need to return to unpopped syntactic ATNs, so as to ensure completeness of parsing, is quite different in motivation from that of returning to an interrupted topic in dialogue processing. In syntactic parsing one must so return, but in dialogue one can sometimes return in a way that is pragmatically inappropriate and we must deal with that below.

3.3 A Modest DAM Proposal

(l) We propose a single pop-push stack architecture that loads structures of radically differing complexities but whose overall forms are ATNs at most (though this power will not always be needed). The algorithm to work such a stack is reasonably trivial and well understood, though below we will need to suggest one amendment to the classical algorithm so as to deal with a dialogue revision problem that cannot be dealt with by structure nesting.

(m) The argument for such a structure is its combination of power, simplicity and perspicuity (we used it in the simple mini-CONVERSE system designed for an interactive child's toy). Its key language-relevant features (known back to the time of Woods (Woods, 1970) in practical parsing) are:

(n) the fact that structures can be pushed down to any level and re-entered via suspended execution allows nesting of topics as well as features like barge-in and revision with a smooth and clear return to unfinished materials and topics. This is so well known that it has entered the everyday language of computer folk as "stack that topic for a moment". The note of caution is that, although in recursive syntax, incomplete parsing structures must be returned to and completed, in dialogue not all incomplete structures should be re-entered for completion.

(o) the ATN structures (to be stacked) have Turing Machine power, and are subject to no limitations of a finite state sort: i.e. a command on a transition arc can do anything from filling a data slot to causing a new structure to be pushed; they can naturally be seen as "update rules" in terms of the systems discussed earlier, which suggests that 'Dialogue Move Engine' may be little more than a renaming of conventional structures. They are also perspicuous to write and understand (given our ability in our Chinese room to make any notes on our structures we like).

Turing Machine formal power is a diversion here: it may well be the case that, for efficiency, some later COMIC should have all its structures reimplemented as finite state rules or graphs. But that is irrelevant here, since such structures, though easy to write, are hard to understand in the absence of any context dependence.

(p) There will be such ATNs corresponding to each of the system-driven sub-tasks (i.e. form filling–the form the bathroom salesman aims to end up with at the end of a client session) which are for information eliciting (and whose ATN commands write direct to the output database), as well as ATNs for standard Greetings and Farewells, and for complex tasks like revisions and responses to conversational breakdowns. Mini-ATNs will express simple dialogue act pairs (such as QA) which can be pushed at any time (from user initiative) and will be exhausted (and popped) after an SQL query to the bathroom database.

We propose that the stack be preloaded with a (default) ordered set of system initiative nets, with Greeting at the top, Farewell at the bottom and

such that the dialogue ends with maximum success when all these and all the information eliciting networks have been popped. This would be the simplest case of a maximally cooperative user with no initiative whatever; he may be rare but must be catered for if he exists!

An obvious problem arises here (noted in earlier discussion), which may require that we adapt the overall DAM control structure:

(q) If the user proposes an information eliciting task before the system does (e.g. the client wants to discuss tile-colour-choice before that structure is reached in the stack) then that structure must be pushed into the stack and executed till popped, but obviously its homologue in the stack must not be executed again when it reaches the top. The integrity of the stack algorithm needs to be violated only to the extent that any task-driven structure at the top of the stack is only executed if the relevant part of the database is empty.

However, a closely related, (indeed inverse) issue (and one that caused the WITAS researchers to change their DAM structure, wrongly in our view) is the situation where a user-initiative forces the revision/reopening of a major topic already popped from the stack; i.e. the user chooses pink tile but later, and at her own initiative, decides she would prefer blue and brings up the topic again. This causes, our proposal, no problems: the tile-colour-choice structure is pushed again (empty and uninstantiated) but with an entry subnetwork (no problem for ATNs) that can check the database, see it is non-empty, and begin the subdialogue in a way that shows it knows a revision is being requested. It seems clear to us that a simple stack architecture is proof against simple arguments based on the need to revisit popped structures.

A similar device will be needed when a long dormant, partly executed, net on the stack is reentered after a long delay; a situation analogous to a very long syntactic dependency or long range coreference. In such cases, a user should be asked whether he wishes to continue (to completion) the suspended network. This will require, at every network node, a loop that checks some timer and asks a confirming question if the time lag since the execution of the preceding node is too great, at which point it could respond with a prefix like "I know it's been a while but should we finish up the discussion of <net_name>"?

What has not been touched upon here is the provision, outside the main stack and content-structures, of DAM modules that express models of the users goals/beliefs/intentions and which reason over these. We shall postpone this as inessential for getting a DAM started provided what we ultimately propose can transition from simpler to more complex structures and functions without radical redesign. At this stage, we do not believe anything as complex as the ViewGen system (Ballim and Wilks, 1991) will ever be required for normal applications. To use it for bathroom advice would require an implausible scenario where the advisor has to deal e.g. a client couple, possibly talked to separately so that the system has to construct a couple's views of each other's wishes.

We do not believe we should start by taking account of the needs of this scenario, fun though it would be for research. However, we will expect to build into the DAM some explicit representation of plan tasks, and this will give no problem to an ATN since recursive networks can be, and often have been, a standard representation of plans, which makes it odd that some soi-disant radical redesigners of DAM's have argued against ATNs as DAM models, wrongly identifying them with low-level dialogue grammars, rather than, as they are, structures (ATNs) more general than those for standard plans (RTNs). It is worth remembering here that twenty-five years ago, Schank modelled his text-processing scripts on the SRI planning structures of STRIPS, of which they were no more than a "linguistic version". It is time to bring the modelling of planning and dialogue modelling back together, long after they were split apart by the plan-analysis approach to dialogue of Allen et al. (Allen et al., 1995)

4 How the ATNs are Created

At the moment the ATNs are created manually using expert knowledge of the application domain. Since our application task is goal-driven, we produced a schema of sub-tasks that the user will address in order to complete the final task (in this case, designing a bathroom). The sub-tasks pro-

vide an inherent modular structure for the creation of ATNs. Given the conversational nature of the interface, the user may stray from the intended task and sub-tasks. For such cases we will be building general purpose ATNs that capture the gist of the utterance, and try to give a reasonable response. To do this, we may use a general resource such as EuroWordnet to capture the essence of the topic words that are not part of our application domain and interpret them in a context taken from hypernyms of Wordnet (Miller (Ed.), 1990). We will, of course, then try to steer the user back to their point of departure from the application task.

As was explained earlier, we will start the program with a stack of ATNs that represent the tasks/sub-tasks that the user needs to satisfy in order to complete the overall task of our application. When our program is using system-initiative to drive the program, we will know which ATNs to run to fulfil each particular sub-task. Due to the stack structure, we also know the order of ATNs to process. However, like most user-friendly systems today, this system is mixed-initiative so that the user may drive the order of tasks (or non-tasks) at any point. It is this issue that brings us to consider how we will select the most relevant ATN in our system in order to address the user's utterance.

When we are in the greeting ATN and prompting for the users' name, we will expect a name as input. If, in return, we get an utterance without a name, then we must scan for change of topic. we can only be sure that we don't have a name if we have a category such as NAME with a list of all possible names to check against. Likewise, we will need categories for COLOUR and STYLE which will contain all members of each category. But of course, it is not enough to recognise that a member of a class is part of the user's utterance, because it may be a negation - 'I don't like the classical style' or 'show me anything but the country style'

Each ATN has associated with it a list of topics and events. Every arc of each ATN has associated with it a category which indicates the type of response we are expecting to receive. The categories for each arc of an ATN taken all together give an indication of the function/meaning of that ATN. This gives the ATN an identifier which we can use in deciding which ATN is best suited to a particular user utterance.

Examples of our ATNs are:

- Bathroom Styles ATN- allows a user to choose one of 4 general bathroom categories
- Bathroom Style Projects ATN- allows a user to choose one of six projects done in a particular bathroom style
- Bathroom Measurements ATN - allows a user to input their own bathroom measurements. This ATN has sub-ATNs for getting the different parts of the bathroom - wall vs. floor etc.

4.1 The COMIC Ontology

The COMIC Ontology serves two purposes. It links the user terminology to the concepts used in the COMIC application. For example, users may refer to a toilet (concept toilet) using different lexical items - toilet, loo, john, comode, etc. It also links the Bathroom application specific Database (ViSoft) to the concepts in the COMIC application; thus style and colour features stored for different kinds of sanitary ware in the Database are linked to the COMIC directly to the COMIC Ontology.

The COMIC Ontology will be built manually using corpus-based studies of user interactions with bathroom design applications as well as the existing bathroom database classification scheme.(ViSoft). The corpora will be gathered through a set of WOZ experiments that simulate our application capturing the language, style and multimodality of the users.

4.2 Identifying the Most Appropriate ATN for an Utterance

The ATNs are selected using the COMIC Ontology and the semantic information extracted during the language processing module (Fusion).

There will be a list of words associated with each of the concepts in the COMIC Ontology. This will be done in an early stage of processing. An example concept is sanitary ware and the associated lexical items are bathroom furniture, bathroom furnishings, bathroom china, etc. It is important to note that the lexical items are not only

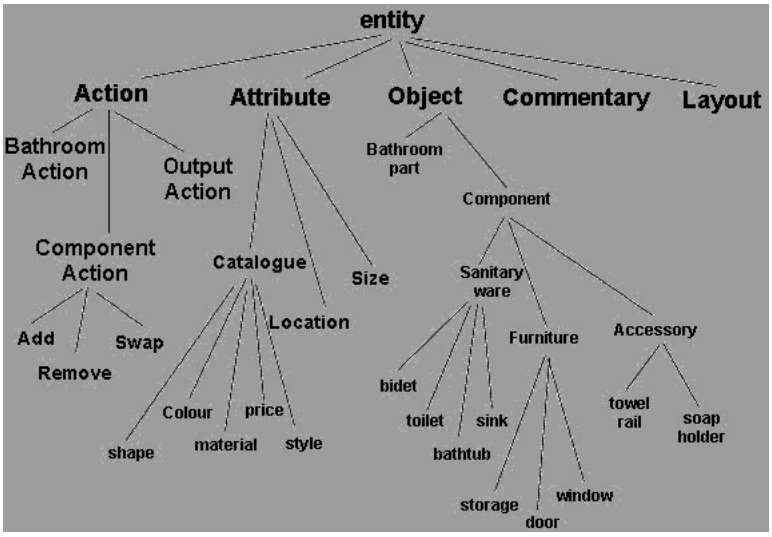

Figure 2: Sample COMIC Ontology

single words, but also include phrases with multiple words. We will identify significant words and phrases using a corpus-driven approach. Using WOZ experiments, we will identify the relationship between concepts in our ontology and lexical items closely linked to such concepts. Most of the nodes in our ontology signify nouns, which although contentful, are not all we need to establish the context of the user's utterance. What the user wants to do with the bathroom object is necessary information that we need in order to establish which ATN should be accessed at any given time.

But it is not enough to index/select the ATNs by keywords alone. There are many activities/tasks that can be achieved given any particular keyword. For example, given the topic of tiles, a user may want to 1) change the colour, change 2) size of the tiles, 3) swap for completely different tiles etc. This shows that the activity or task that the user wants to perform also acts to narrow down the context of an interaction.

As we said earlier, this interaction is a task-based scenario, where the user undertakes a series of subtasks to complete the overall task of our application (bathroom design). As such, we will refer to the tasks as events and have listed below an example list of events that will be used in our domain.

The information returned from the language processing module (Fusion) will be put into the form of subject-verb-object triples. Through this formalism, we can decide how to index the list of ATNs that we have.

Examples:

- I like this bathtub
  `Subj (user) verb (like) obj(bathtub)`
- Show me the classical style
  `Subj(system) verb(show) obj (classical style)`
- Show me project 5
  `Subj (system) verb (show) obj (project 5)`
- Please exchange the blue tiles for white ones.
  `Subj(system) verb(exchange) obj (blue tiles) misc( white tiles)`

We will have a pre-defined set of bathroom events which correspond to verbs. We will enrich the list of lexical items associated with each event using a thesaurus-like knowledge base called EuroWordnet. Below, there is a list of bathroom events with their corresponding lexical items taken from Wordnet 1.6 (same as the English part of EuroWordnet) . It is worth noting that since EuroWordnet is a multilingual resource, we have a way of associating multilingual lexical items to our bathroom events and objects

Examples:

- show_event: show, display, list, present, reveal etc.

- exchange_event: exchange, substitute, swap,
- arrange_event: arrange, put, set, place, position
- choose_event: choose, prefer, like, select, pick out

4.2.1 Changing Topic or Domain

The issue of changing topics is a very serious one in any dialogue system. Given that a user can change topic at any point in the conversation, a human-computer system must always be scanning for topic change. Although handling topic change outside a domain is not fundamentally different from handling topic change within a domain, the former requires a potentially limitless amount of knowledge and therefore has strong practical implications. Our approach will handle topic shift within a domain by attaching keywords and events to topic areas (implemented as ATNS), but will not handle topic shift outside of our application domain except to try to steer the user back to the task at hand.

5 Conclusion

Whether a dialogue management system can be completely general is a debatable point, but we have presented a system that is aimed to that end as much as possible. This was done by separating the application specific knowledge from the control structure allowing for maximum reusability across applications. We have shown how our system is integrated into the EU COMIC project and we have explained our choice of using well known structures like ATNs and stacks in an innovative way to support flexible next generation internet applications. In the future we intend to test the reuseablity issue in two other dialogue interface applications - a career/job matching system and a personal/email organiser system.

6 Credits

Many thanks to the COMIC partners: Max Planck Institute for Psycholinguistics, Nijmegen, The Netherlands; Max Planck Institute for Biological Cybernetics, Tübingen, Germany, University of Edinburgh, Division of Informatics, United Kingdom; Deutsches Forschungszentrum für Künstliche Intelligenz GmbH (DFKI), Intelligent User Interfaces, Saarbrücken, Germany; Katholieke Universiteit Nijmegen, Nijmegen Institute for Cognition and Information, The Netherlands; ViSoft, Sindelfingen, Germany.

References

J.F. Allen and C.R. Perrault. 1980. Analyzing Intentions in Utterances. *Artificial Intelligence*, 15(3):143–178.

J.F. Allen, L.K. Schubert, G. Ferguson, P. Heeman, C. Hee Hwang, T. Kato, M. Light, N.G. Martin, B.W. Miller, M. Poesio, , and D.R. Traum. 1995. The TRAINS Project: A case study in building a conversational planning agent. *Journal of Experimental and Theoretical AI (JETAI)*, 7:7–48.

A. Ballim and Y. Wilks. 1991. *Artificial Believers*. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

J. G. Carbonell. 1979. Towards a Self-Extending Parser. *17th Meeting of the Association for Computational Linguistics*, pages 3–7.

K.M. Colby. 1971. Artificial Paranoia. *Artificial Intelligence*, 2.

R. Cooper, S. Larsson, C. Matheson, M. Poesio, , and D. Traum, 1999. *Coding in Structural Dialogue for Information States : Deliverable D1.1. Trindi Project*.

J.R. Hobbs. 1993. The Generic Information Extraction System. In *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, pages 87–91. Morgan Kaufman.

O. Lemon, A. Bracy, A.R Gruenstein, and S. Peters. 2001. The Witas Multi-Modal Dialogue System I. *Eurospeech2001*, pages 1559–1562.

D. McDermott, 1981. *Artificial Intelligence Meets Natural Stupidity*. MIT Press.

G. A. Miller (Ed.). 1990. WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312.

W.A. Woods. 1970. Transition Network Grammars for Natural Language Analysis. *CACM*, 3(10).