metalogue

METALOGUE: Deliverable D1.1

Specification of the METALOGUE System Architecture

The METALOGUE Consortium

February 2014

Version N°: 1.1

Main Authors:

B Horan—Editor
P Albert—Contributor
N Campbell—Contributor
V Petukhova—Contributor
C Samuelsson—Contributor
A Stricker—Contributor
J van Helvert—Contributor

Project funded by the European Community under the Seventh Framework Programme for Research and Technological Development

Project ref. no.	ICT – 611073
Project title	METALOGUE – Multiperspective Multimodal Dialogue: dialogue system with metacognitive abilities

Document status	V1.0
Contractual date of delivery	28 February 2014
Actual date of delivery	28 February 2014
Document number	D1.1
Deliverable title	Specification of the METALOGUE system architecture
Dissemination level	Public
Type	Report
Number of pages	21
WP contributing to the deliverable	WP 1
WP / Task responsible	TCD/UESSEX
Contributing partners	Charamel, DFKI, TCD, UdS
Reviewers	Jan Alexandersson, Thomas Kleinbauer.
Author(s)	Bernard Horan, Pierre Albert, Nick Campbell, Volha Petukhova, Christer Samuelsson, Alexander Stricker, Joy van Helvert.
EC Project Officer	Pierre-Paul Sondag
Keywords	METALOGUE, multi-modal, dialogue, metacognition

Table of Contents

FIGURES

TABLES

1. Executive Summary

This document specifies the open architecture of the integrated METALOGUE system as envisaged in the initial stages of the project. The architecture will be further refined and elaborated as the project progresses and this will be reflected in subsequent versions of this document.

The envisaged overall architecture can be viewed as three layers. The low level core layer comprises a system architecture which supports several input and output modalities, such as spoken natural language, facial expressions, body posture and biosensor data, and is designed to be modality-agnostic; a high level layer providing a toolkit for developers (API) to create applications; and finally a validation layer in which the API to is employed to create an application that allows user partners to validate the benefits of the project.

The proposed architecture will follow the principles of multi-modal dialogue architectures described in the literature and will include the following components:

- Sensor specific input
- Interpretation (Modality specific analyses)
- Fusion
- Discourse and dialogue manager
- Metacognition
- Fission
- Generation
- Rendering (Text to speech, visual)

In addition, components may be associated with knowledge bases such as common-sense ontologies and domain-specific structured databases. Communication between the components will use common standards, but the details of the transport mechanisms and protocols will be defined in future iterations of this document.

2. Introduction

This document specifies in detail the open architecture of the integrated METALOGUE system. This system includes components that are necessary for the implementation of the processing stages: dialogue management, speech recognition, speech synthesis, analysis and rendering in other modes, user model, etc.. It also contains components for moving to a new application domain.

This document is subject to revision, as the architecture is informed and refined by the following deliverables:

Deliverables no.	**Deliverables title**
D 1.2	Specifications of initial & extended scenarios
D 1.3	Specifications of experimental design
D 1.4	User requirements and involvement report

Table 1: Deliverables Informing and Refining Architecture

3. Overall METALOGUE Architecture

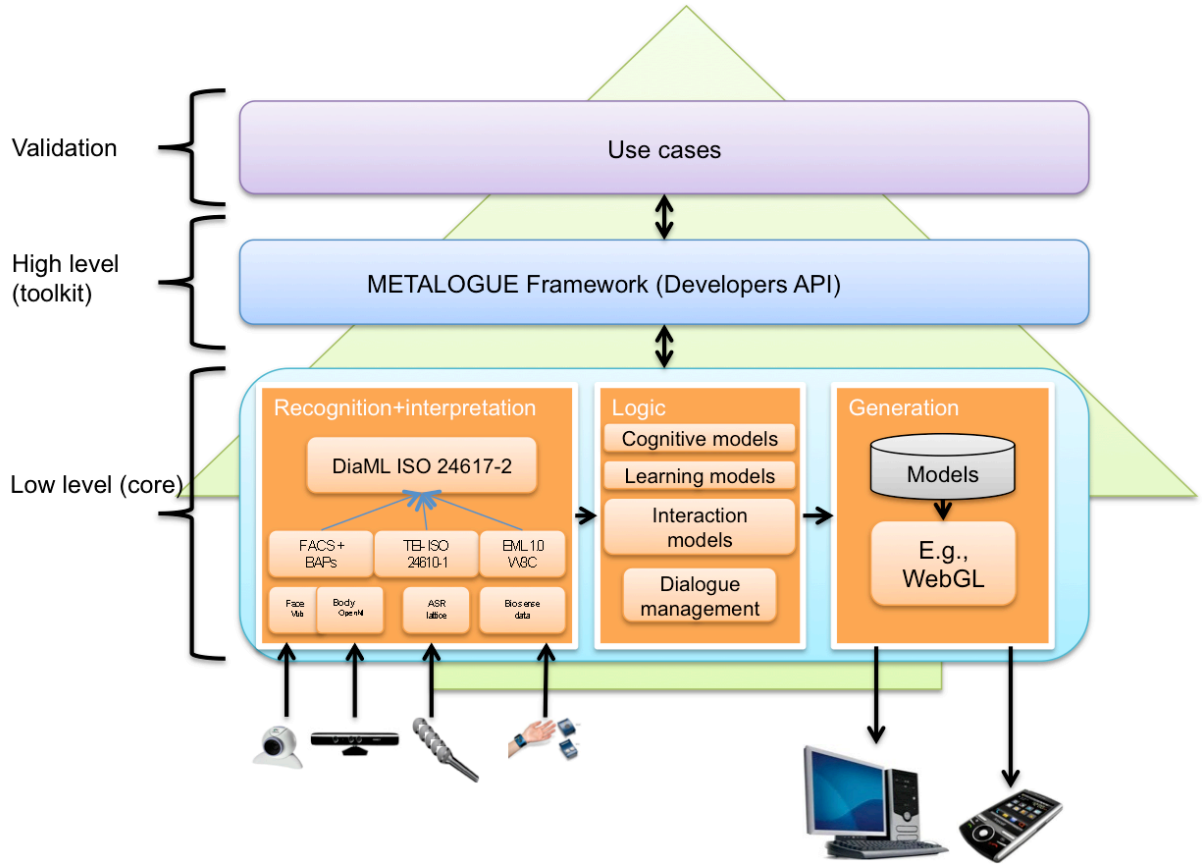

Figure 1: Overall METALOGUE Architecture

3.1 Low Level

The system architecture is manifested as the low level layer of the overall METALOGUE architecture.

The system architecture will include several modalities, such as spoken natural language, facial expressions, body posture and biosensor data. Where appropriate, these modalities are designed to be symmetric: a modality available for user input will have a counterpart for output. For example, the ability to use facial recognition will be complemented by the generation of facial expressions via virtual characters. This approach allows for a natural interaction and is especially useful in multi-party interactions. In such settings, some modalities may be exclusive to two communication partners (for example, speech), but other participants can still interact in parallel, for example, using gestures.

The system architecture is designed to be modality-agnostic as far as practical. Therefore, its aim is to provide APIs that allow developers to connect new devices and interpretation functionality for their sensor data. The only requirement from the architecture, apart from the

APIs, is that the interpretation of the sensor data provides results in a uniform semantically enriched format that can be mapped to DiaML (Bunt et al., 2012) communicative acts.

The cognitive, learning and interaction models are developed in work packages two, three and four, respectively. The models are closely integrated and direct the operation of a dialogue manager component. The dialogue manager acts over a shared information state that incorporates and manipulates information from all models. The dialogue manager is plan-based and works on the level of goals and sub-goals. These take the form of tasks that are generated by the learning model based on the current situational context and the progress of the learning objectives. From this, the dialogue manager devises intentions and character-specific goals for the virtual characters and dynamically plans how the intentions can be achieved by taking into account the available interaction patterns and conditions or requirements arising from the current situational context.

The result of the planning phase is a set of actions for each virtual character. These actions are then rendered in a virtual environment using a combination of generated speech output and gestures, facial expressions and body postures that can be performed by virtual characters in that environment. The data, including user inputs and rendered outputs, is available to all components in a semantically enriched representation, by the discourse module of Figure 2. This is necessary to enable the processing of some interaction phenomena. For example, a virtual character's behaviour must be present and accessible to the interpretation components if they are to be able to resolve a subsequent user input that references that behaviour, whether it is a linguistic reference (for example, an elliptic expression), a deictic reference or a reference of some other kind.

The system architecture is designed to allow conversational interactions in real time and use processing shortcuts where time-critical reactions are desired. For example, if a spoken input is detected, immediate attention feedback via gaze (say) is generated by the virtual characters, bypassing slower full semantic processing of the input. This feature is very important in creating and preserving the perception of immersion and responsiveness for the user. In a further step, the system will feature incremental processing, as the means of achieving such behaviour. Rather than waiting for a complete utterance from the user (such as a sentence), input (such as a word or gesture) is processed in real time, updating system understanding as new information is expressed.

3.2 High level

This layer will include an easy-to-use API that allows developers to create new applications based on the system architecture.

This layer of the architecture acts as an abstraction feature for the low level core and hence depends heavily on the design and implementation decisions for the core. As a consequence,

its development and further specifications will start in a second stage of the project once the low level is mature enough for this API to be relevant.

3.3 Application layer

The application layer will make use of system the architecture to create an application that allows user partners to interface with real-world applications and validate its benefits. The project is designed to go through several incremental iterations that will be used to gradually improve and refine the models for learning, interaction and cognition, the underlying knowledge representation and the general usability of the system. Thus, it is crucial that demonstrator systems are deployed early and accessible to users. In this way, data from user interactions can be collected and evaluated to gain insight with regard to any shortcomings and provide suggestions for improvements for the subsequent development cycles. To this end, the validation layer will provide means to record and collect interaction data together with meta-data such as error rates, task completion and measures of user satisfaction.

4. System Architecture

The system architecture will follow the principles of multi-modal dialogue architectures such as those described in (Herzog & Reithinger, 2006), and as illustrated in Figure 2 below. Furthermore, each of the components of the architecture may be associated with a knowledge base. Examples of knowledge bases include:

- Structured databases, such as those particular to a call-centre.
- Ontologies, such as general 'common sense' ontologies including OpenCyc (Matuszek & Cabral, 2006) and WordNet (Fellbaum, 1999), as well as ontologies particular to the domain.

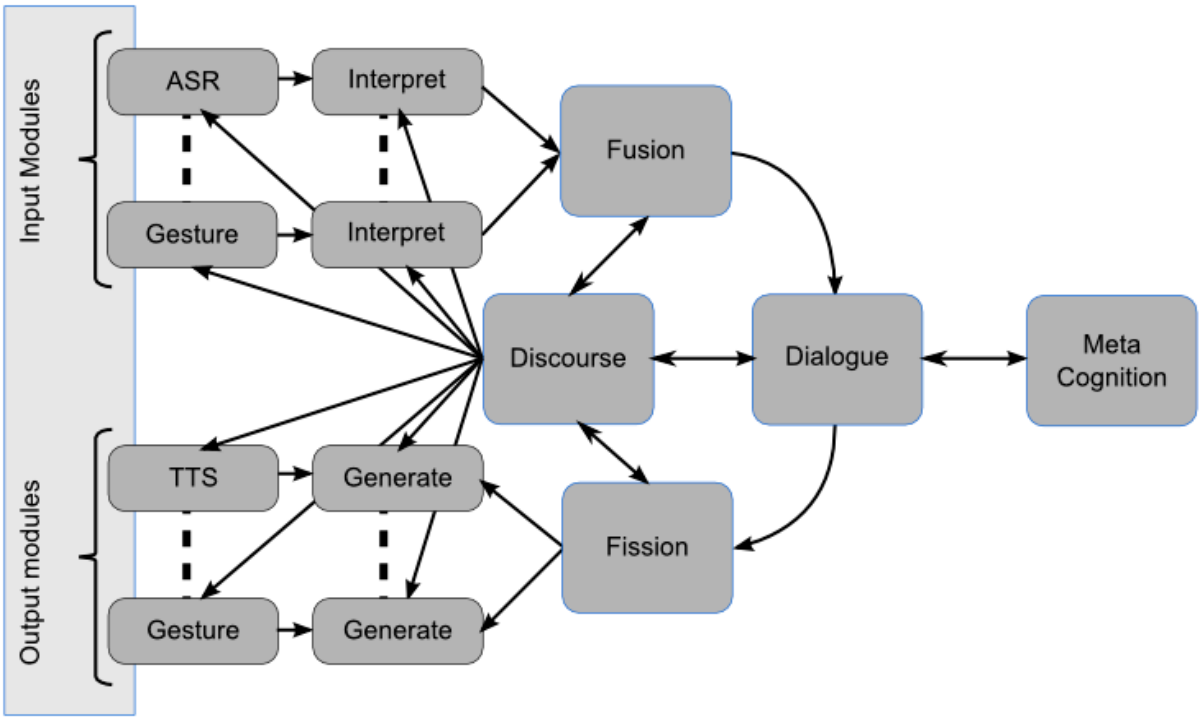

Figure 2: Multi-modal Dialogue Architecture

The components themselves are also described in Work Package five, associated with the following deliverables:

Deliverables no.	Deliverables title
D 5.1	Multimodal input recognition components
D 5.2	Multimodal Interpretation Management component
D 5.3	Multimodal Dialogue Management component
D 5.4	Metacognitive Control Manager

D 5.5	Multimodal Generation Management component
D 5.6	Virtual Agents
D 5.7	Software for visual analytics of learner' behaviour and computer-supported learning tools

Table 2: Deliverables Associated with Components

Communication between the components will use common standards. The components will, where possible, use *de jure* standards as specified by bodies such as the W3C, IETF and IEEE. However, if in the absence of de jure standards, *de facto* standards already exist, the architecture will use those.

The input from each sensor-interpreter pair in the input array is merged by the fusion unit and made available to the discourse and dialogue units. There is a converse rendering portion: the output is broken by the fission into the different output modalities and each modality consists of a generation-realization pair.

The following sections describe each of the components according to their category. We have indicated the Tasks and Work Packages with which they are associated.

4.1 Sensor-specific Input

This category of components is responsible for the identification and recognition of signals emanating from sensors. The specification for this category of components is provided in D5.1.

4.1.1 Automatic Speech Recognition

The development of the Automatic Speech Recognition (ASR) component (T5.1.1) is based on Kaldi[1], an emerging speech recognition research toolkit that is freely available under version two of the open source Apache license[2]. Kaldi emphasises generic algorithms and design (Universal Recipes™), and is thus distributed without acoustic and language models. Therefore, these models will be developed using domain-specific data as part of the METALOGUE project.

The output from Kaldi is in the form of lattices and sentences, examples of which can be found on the Kaldi website.

[1] http://kaldi.sourceforge.net
[2] http://www.apache.org/licenses/LICENSE-2.0

4.1.2 Visual Movement Tracking

This component provides face and eye gaze tracking and facial expression encoding (T5.1.2), body tracking and body expression encoding (T5.1.3), and temporal segmentation and recognition of static and dynamic face and body expressions (T5.1.4).

This component will employ a Kinect sensor—it includes a camera for a video feed; an infrared projector and a sensor for 3D positioning; and a microphone array for speech and high level audio processing, cf Section 4.1.1. The Microsoft Developer Network provides a Software Development Kit[3] (SDK) for the Kinect, which describes the API in detail. A JavaScript API[4] is available, using an event-driven paradigm.

The output from the Kinect API includes the following:

- **Body tracking**: matrices (orientation and position) and tracking result.
- **Face tracking**: mouth, eyes, eyebrow, open/close eyes, noise, and tracking result.

The tracking output data may be encoded using XML-based structures, such as the Text Encoding Initiative (ISO 24610-1:2006 Language resource management—Feature structures—Part 1: Feature structure representation[5]). Other encoding schemes will be examined as part of the project, such as:

- Body and postures: The Body Action and Posture coding system (Dael, Mortillaro, & Scherer, 2012).
- Hand gestures: The Hamburg Sign Language Notation System—HamNoSys (Hanke, 2004).
- Facial expressions and associated emotions: Facial Actions Coding scheme (Essa & Pentland, 1997).

These schemes have been designed to support the characterization of movements in terms of low-level (or surface) behavioural features, such as changes in muscular activity (body parts involved and form of movement), or direction, trajectory and speed of movements. For example, HamNoSys is a transcription system for coding hand gestures by describing shape, direction, speed, length and form of movement, hands orientation and location.

4.2 Interpret: Modality–Specific Analysis

The components that take on the role of interpretation are described in T5.2 as the Multimodal Interpretation Manager. The specification for this category of components is contained in D5.2.

[3] http://msdn.microsoft.com/en-us/library/jj131023.aspx
[4] http://msdn.microsoft.com/en-us/library/dn435664.aspx
[5] http://www.iso.org/iso/catalogue_detail.htm?csnumber=37324

4.2.1 Natural Language Processing

The Natural Language Processing component takes its input in the form of word lattices from the ASR component, encoded using an XML standard format yet to be determined.

The component will provide output as a slot-value pair (or n-tuple) semantic representation encoded using an existing XML standard format. No decision has yet been made on this format, however, the W3C Speech Recognition Grammar recommendation[6] is a potential candidate**.** The output will be communicated to other components using asynchronous message passing, cf. Section 4.9.

The component will provide parallel processing pipelines for grammar-based and data-driven interpretation of the input. These results will be merged within the component, possibly using priority merge. In subsequent stages of the project, initial grammar processing may be merged with the ASR component using compile-time transducer composition.

4.2.2 Gesture and Facial Interpretation

This component takes the tracking movement data (from section 4.1.2) and produces as output a semantic interpretation of gesture and head movements, postures and facial expressions. By 'semantic interpretation', we mean an assignment of a meaning in terms of *type* of movement. For example, an up and down head movement is a *nod*, a left to right head movement is a *head shake*, and elongating the lips and lifting the lip corners is a *smile*. The annotation of the type of tracked movements will allow the determination of variations in bodily activity that may have one and the same meaning (such as different spatial, temporal, durational and intensity qualities). This information will provide an important empirical basis for precise further semantic or pragmatic analysis, for example to establish whether similar *types* of movement containing different low-level characteristics have the same communicative function.

There are many existing XML-based encoding schemes for visible body movements/actions. The project will use the MUMIN scheme (Allwood, Cerrato, & Dybkjaer, 2005) combined with the scheme provided with the ANVIL tool (Kipp, 2003) in addition to some extensions. This combination of schemes will enable the project to encode 84 movement types, as follows:

- two for gaze (averted versus direct);
- nine for head movements (such as, head nod, jerk, shake);
- 40 for hand and arm movements (such as, pointing, purse, pray);
- 24 for facial expressions (such as, interest, happy face, puzzled, surprise); and

[6] http://www.w3.org/TR/speech-grammar/

- nine for posture shifts (for example, working position, learning forward, backward, random posture shifts).

4.3 Fusion

This component is responsible for combining the modality-specific analyses into a fused representation of a dialogue action encoded using the DiAML format. A dialogue act is a key notion in theories of dialogue. Dialogue acts are often used in studies of dialogue phenomena: they provide a way of describing the interpretation of communicative behaviour of participants in a dialogue, and in the design of dialogue systems. Informally speaking, a dialogue act is an act of communicative behaviour performed for some purpose. For example a dialogue act may be used to provide information, to request the performance of an action, to apologise for a misunderstanding, or to provide feedback. ISO standard 24617-2[7] defines a dialogue act as:

> *communicative activity of a participant in dialogue, interpreted as having a certain communicative function and semantic content*

A *communicative function* specifies the way semantic content is to be used by the addressee to update the addressee's context model when the addressee understands the corresponding aspect of the meaning of a dialogue utterance. In other words, we can call this the *speaker's intention*. The interpretation of dialogue behaviour is primarily based on the recognition of the speaker's intentions.

We will explore a hybrid approach to the specification of this component that combines feature, chunk and interpretation level fusion. Input processing at the feature level will be available due to the low-level features obtained from all available information sources from multiple modalities, including:

1. linguistic information derived from the surface form of an utterance: lexical and co-locational information;
2. perceptual information from multiple modalities available to dialogue participants, including acoustic and prosodic properties of utterances as well as information from visual and other modalities;
3. contextual information obtained from the preceding dialogue context and dialogue structure, as well as global context properties like dialogue setting, knowledge about dialogue participants, and domain knowledge.

This output from this component will be in the form of hypotheses on dialogue acts with confidence scores.

[7] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=51967

An example output from this component is provided in the Annex (section 5.2).

Figure 3 below summarises the workflow for this component: feature extraction, selection and dialogue act classification.

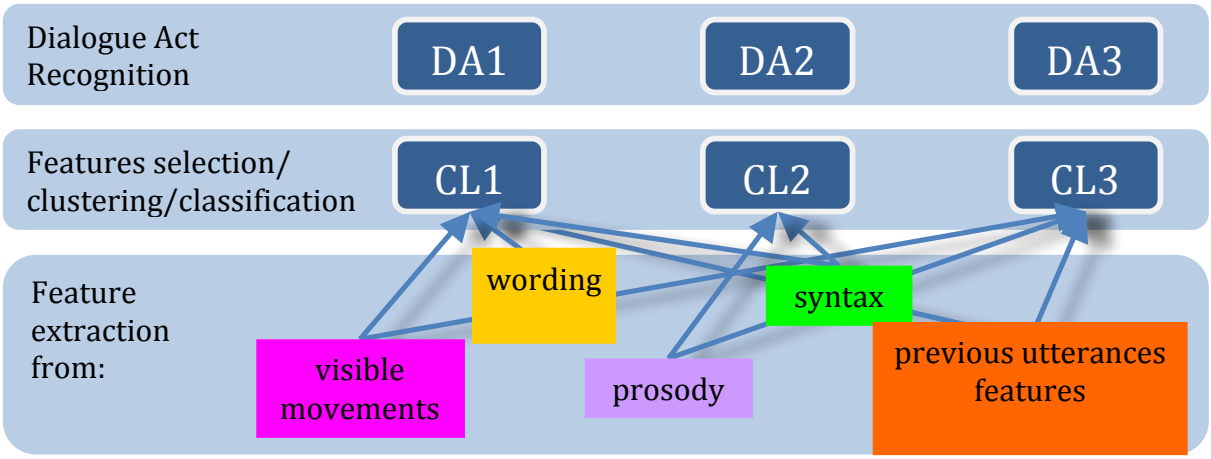

Figure 3: Workflow Summary for Fusion Component

4.4 Discourse and Dialogue Manager

The METALOGUE discourse and dialogue manager has a complex internal multi-agent architecture consisting of several sub-components: a Discourse Manager that is constantly updated; a Dialogue Manager consisting of Task Manager/Planner and Interaction Control Agents; and various Metacognitive Control/Evaluation Agents. Task 5.3 describes the requirements of the dialogue manager, and D5.3 contains the specification for this component. Figure 4 below illustrates its proposed architecture.

The input for this component is a fused representation from the Modality Fusion component. This comprises output from Multimodal Dialogue Act recognizer plus output from Interpretation Manager, represented in terms of slot values in frames. As output, this component provides candidate dialogue acts for generation, in the same format as that produced the Interpretation Manager.

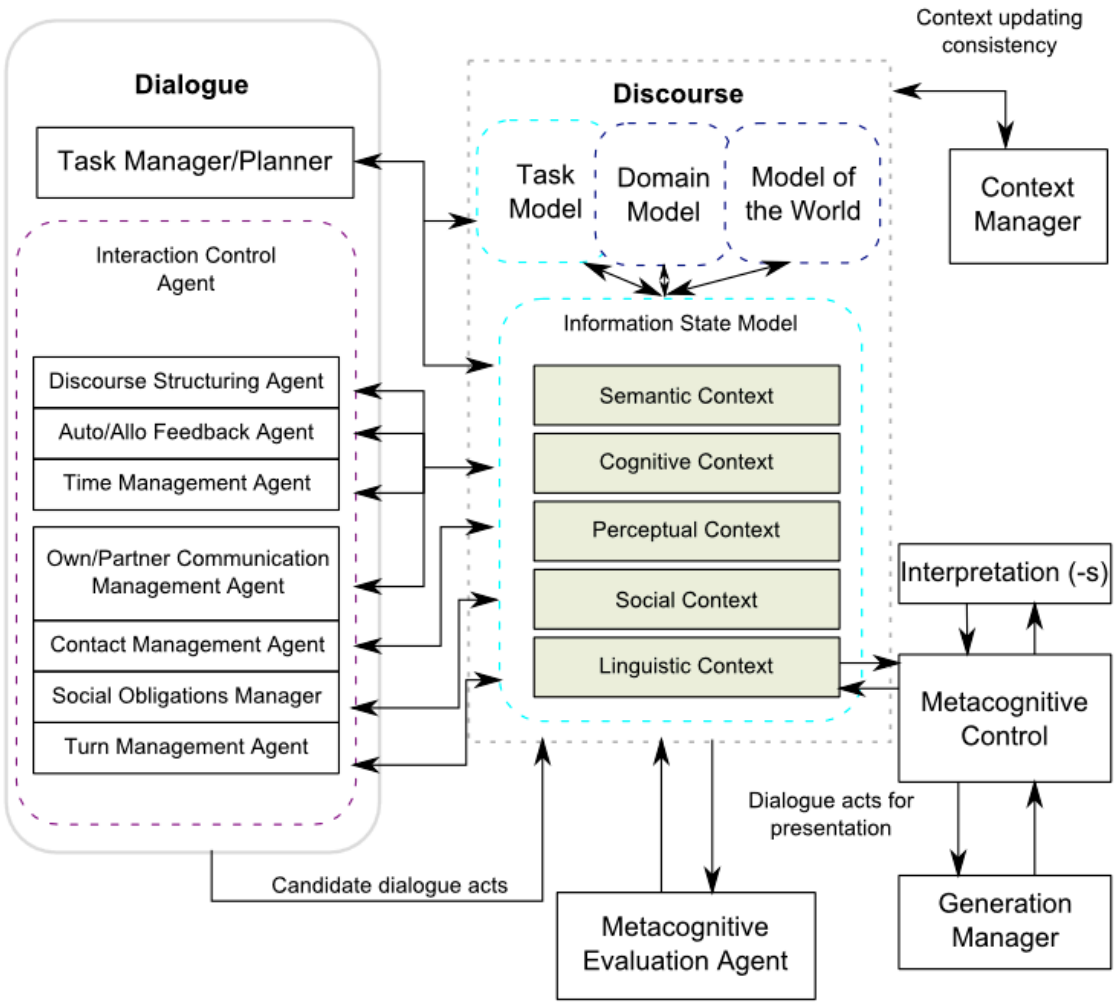

Figure 4: Discourse and Dialogue Manager Architecture

The role of the Discourse Manager is to manage the state of its models, in response to recognised dialogue act inputs. The Metacognitive Control Agent is responsible for taking care of selecting/filtering updates. The discourse models contain all the information considered relevant for interpreting dialogue utterances and for generating dialogue acts. This model includes information concerned with

1. A model of the dialogue task (hierarchical—partly static)
2. A model of the domain (domain knowledge, such as domain-specific ontologies—mostly static)
3. A model of the world (world knowledge, that is, common sense facts—static)
4. Information State (dynamically updated by every incoming dialogue act) that consists of:
    i. the participants' information about the underlying **task** and its domain, as well as their beliefs about the dialogue partner's information of this kind (*semantic context*);

ii. the participants' **state** of **processing** (*cognitive context*);

iii. the availability and properties of communicative and **perceptual channels**, and the partner's presence and attention (*physical/perceptual context*);

iv. **communicative obligations** and constraints (*social context*);

v. the preceding and latest dialogue contributions, and possible immediate **dialogue plans** (*linguistic context*).

Furthermore, the Dialogue Management Agents (**Task** and **Interaction Control**) continuously monitor the discourse model and, if appropriate, attempt to generate candidate dialogue acts according to their associated role/task.

The **Metacognitive Evaluation Agent** monitors the list of candidates and decides which of them can be combined into a multifunctional utterance for generation, and when. Some of the dialogue act candidates may have higher priority and should be generated at once, some may be stored for possible generation in later user interactions (or 'turns'), and some will already be implicitly performed through the performance of other candidate acts. The process of combining and evaluating dialogue acts for generation involves logical, strategic, and pragmatic considerations, and allows for the flexible implementation of different dialogue strategies and styles of communication.

In the first demonstrator, the system will assume an observational/critiquing role, rather than that of an interlocutor, and the generation portion will not be relevant.

4.5 Metacognition

The metacognition functionality of the architecture is embodied in the metacognitive control manager component, as described in Task 5.4, and specified in D5.4. The role and behaviour of metacognition are major research aspects elements of the project.

There are two types of Metacognitive Control/Evaluation agent (see Section 4.4). The **Metacognitive Control Agent** (MCA) is responsible for selecting/filtering updates after/during *interpretation*. The **Metacognitive Evaluation Agent** monitors the list of candidates and decides which of them can be combined into a (multimodal) system utterance(-s) for *generation*, and when. Both agents operate in terms of dialogue acts and output a selected/filtered list of dialogue acts.

The Metacognitive Control Agent also monitors and controls *processing resource allocation* of the different modules. Resource allocation depends heavily on the content. The agent analyse the nature of the input and evaluate the allowed - or expected - duration of the reflexion accordingly. The MCA also manages resources according to dialogue requirements, e.g. enforcing time limit to comply with reactivity, generating feedback as a natural reaction to acknowledge input.

4.6 Fission

The multi-modal fission component (described in Task 4.2.5) is tasked with deciding how to distribute the communicative act to be performed by the system into the available modalities. For instance, when using both speech output and the virtual human animation engine, a "positive feedback" act may be presented with animation only (nodding), with speech only ("yes") or by a combination of both. The output from the dialog manager is passed as input to this component, and can potentially consist of a list of multiple dialogue acts.

This component is responsible for making the decision about which dialogue acts to output to the Generate component as well as how to divide the chosen dialogue acts to be delegated to the separate modalities. This decision may be informed by a meta-cognition agent. In addition to mere separation, the fission component is also responsible for coordinating and synchronizing the different modalities, for example, so that gesture and speech or lip movement are rendered synchronously.

4.7 Generate

The dialog act information delegated to the different modalities by the fission component (described in section 0) is then transformed by the *generation* component into a representation suitable for the chosen output modalities. The generation component thus takes as input the part assigned to it in dialog act representation and generates from it a native data format to be passed to the subsequent rendering components. This is ideally a one-to-one translation that does not require further decision making. However, if the generation into native format fails, perhaps because of the lack of required resources or user interruption, this failure can be communicated to the fission component for it to produce alternatives. This approach gives an additional level of robustness to the system. As with other components, the data exchange formats for the inter-component communication will be based on XML.

Task 5.5 describes the elements of this component, and the component is specified in the Multimodal Generation Management component of D5.5.

4.8 Rendering

The rendering components of the system architecture are currently limited to two: a text-to-speech (TTS) component and a visual environment containing a virtual character represented as an avatar. We discuss each in the following sections.

4.8.1 TTS

We have yet to decide on a speech synthesizer implementation for this component; however, it is likely that we will choose one of the several speech synthesizer implementations that are

available to us. These include open source implementations, including Festival[8] and Mary[9], as well as commercial implementations such as Acapela[10], Ivona[11], Nuance[12] and CereProc[13]. The selection of TTS implementation is likely to be dependent on its suitability (and compatibility) for use with the visual rendering component, described below.

4.8.2 Visual

The visual rendering component will take the form of a virtual character rendered using a virtual human animation engine, described in Task 5.6 and specified in D5.6.

Charamel offers the 3D real time animation software *CharActor*[14] specialized for the usage of interactive avatars. The SDK offers interfaces for controlling the animation and speech of the avatars in real time. The behavior and speech of the avatar will be controlled by XML commands received from the generate component.

No decisions have yet been made on the form that the virtual characters will take, but it is anticipated there will be at least one male and female avatar.

The virtual human animation engine will be combined with a TTS implementation. The choice of implementation is constrained by the need to ensure lip synchronization between the avatar and generated speech, via viseme recognition.

4.9 System communication

There is currently a debate in the project whether there be a central controlling unit or whether the various modules and components be autonomous agents. From a low-level perspective, both alternatives can be implemented using a Rendezvous-style message-passing framework. There is a slight complication with the architecture of Figure 4, which presumes black-board communication, but these details will be ironed out during the project.

[8] http://www.cstr.ed.ac.uk/projects/festival/
[9] http://mary.dfki.de/
[10] http://www.acapela-group.se/
[11] http://www.ivona.com/en/
[12] http://www.nuance.com
[13] https://www.cereproc.com/
[14] http://www.charamel.com/produkte/charactor_development_program.html

5. Annex: Additional Descriptions of Software

5.1 Kaldi

Kaldi can be installed on most types of Linux machine, but it has also been tested on Darwin and on Cygwin. It is dependent on the subversion and wget packages to download it and install it.

It also requires the following packages: OpenFst, IRSTLM, sph2pipe, cslite, ATLAS and CLAPACK. If these are not present, Kaldi will automatically download and install them.

The performance of an ASR is measured in Word Error Rate as well as Sentence Error Rate. Kaldi was extensively tested on several corpora, including WSJ, RM, TIMIT, TIDIGITS, PG, and BABEL. It has been shown to achieve state-of-the-art performance.

5.2 Dialogue Manager

An example of interpreting user interactivity and classification as a dialogue act.

Participant 'p2' turning his gaze to participant 'p1' (*gaze1*) and then averting it (*gaze2*), while producing the speech segment *Maybe* (vec2), performing a shoulder shrug (*hag1*), constricting the forehead muscles (*fh1*), raising eyebrows (*brow1*), widening the eyes (*eye1*), lowering the lip corners (*lip3*) and raising the chin (*chin1*), can be segmented as

```
<fs type="functionalSegment" xml:id="fs2">
<f name="verbalComponent" fVal="#vec2"/>
<f name="gazeComponent" fVal="#gaze2"/>
<f name="gestComponent" fVal="#hag1"/>
<f name="headComponent" fVal="#head1"/>
<f name="forehComponent" fVal="#fh1"/>
<f name="eyebrComponent" fVal="#brow1"/>
<f name="eyesComponent" fVal="#eye1"/>
<f name="lipsComponent" fVal="#lip3"/>
<f name="chinComponent" fVal="#chin1"/>
</fs>
```

And classified as dialogue act 'da2' in reaction to dialogue act 'da1' :

```
<diaml xmlns:="http://www.iso.org/diaml/">
<dialogueAct
    xml:id="da1" target="#fs1"
    sender="#p1" addressee="#p2"
    communicativeFunction"="suggestion"
    dimension="task"/>
<dialogueAct
    xml:id="da2" target="#fs2"
    sender="#p2" addressee="#p1"
    communicativeFunction="addressSuggestion"
    dimension="task"
    functionalDependence="#da1"/>
</diaml>
```

6. Bibliography

Allwood, J., Cerrato, L., & Dybkjaer, L. (2005). The MUMIN multimodal coding scheme. *NorFA* …. Retrieved from http://www.ling.helsinki.fi/kit/2006k/clt310mmod/MUMIN-coding-scheme-V3.3.pdf

Bunt, H., Alexandersson, J., Choe, J.-W., Fang, A. C., Hasida, K., Petukhova, V., … Traum, D. R. (2012). ISO 24617-2: A semantically-based standard for dialogue annotation. In *LREC* (pp. 430–437).

Dael, N., Mortillaro, M., & Scherer, K. R. (2012). The Body Action and Posture Coding System (BAP): Development and Reliability. *Journal of Nonverbal Behavior*, *36*(2), 97–121. doi:10.1007/s10919-012-0130-0

Essa, I. A., & Pentland, A. P. (1997). Coding, analysis, interpretation, and recognition of facial expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(7), 757–763. doi:10.1109/34.598232

Fellbaum, C. (1999). *WordNet*. Retrieved from http://onlinelibrary.wiley.com/doi/10.1002/9781405198431.wbeal1285/full

Hanke, T. (2004). HamNoSys-representing sign language data in language resources and language processing contexts. *LREC*. Retrieved from http://www.researchgate.net/publication/228963927_HamNoSysRepresenting_sign_language_data_in_language_resources_and_language_processing_contexts/file/504635281dc21aee6a.pdf

Herzog, G., & Reithinger, N. (2006). The SmartKom Architecture: A Framework for Multimodal Dialogue Systems. *SmartKom: Foundations of Dialogue Systems*, 55–70. doi:10.1007/3-540-36678-4

Kipp, M. (2003). Anvil video annotation system. Retrieved from http://scholar.google.co.uk/scholar?q=kipp+ANVIL&hl=en&as_sdt=0%2C5&as_ylo=2001&as_yhi=2004#3

Matuszek, C., & Cabral, J. (2006). An Introduction to the Syntax and Content of Cyc. *AAAI Spring Symposium:* …. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.68.1357&rep=rep1&type=pdf