CLASSiC

D3.1: Shared Context Model (XML Schema)

Oliver Lemon, Olivier Pietquin, Hervé Frezza-Buet
Verena Rieser, Xingkun Liu,
Philippe Bretier,
Steve Young and James Henderson

Distribution: Consortium

CLASSiC
Computational Learning in Adaptive Systems for Spoken Conversation
216594 Deliverable 3.1

February 2009

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

The deliverable identification sheet is to be found on the reverse of this page.

Project ref. no.	216594
Project acronym	CLASSiC
Project full title	Computational Learning in Adaptive Systems for Spoken Conversation
Instrument	STREP
Thematic Priority	Cognitive Systems, Interaction, and Robotics
Start date / duration	01 March 2008 / 36 Months

Security	Consortium
Contractual date of delivery	M12 = February 2009
Actual date of delivery	February 2009
Deliverable number	3.1
Deliverable title	D3.1: Shared Context Model (XML Schema)
Type	Report
Status & version	Final draft 2.0
Number of pages	22 (excluding front matter)
Contributing WP	3
WP/Task responsible	WP3, leader Supelec
Other contributors	
Author(s)	Oliver Lemon, Olivier Pietquin, Hervé Frezza-Buet, Verena Rieser, Xingkun Liu, Philippe Bretier, Steve Young, and James Henderson
EC Project Officer	Xavier Gros
Keywords	Interfaces, Dialogue Context, XML, DTD, Schema

The partners in CLASSiC are:

University of Edinburgh HCRC	EDIN
University of Cambridge	UCAM
University of Geneva	GENE
Ecole Superieure d'Electricite	SUPELEC
France Telecom/ Orange Labs	FT

For copies of reports, updates on project activities and other CLASSiC-related information, contact:

The CLASSiC Project Co-ordinator
Dr. Oliver Lemon
School of Informatics, Edinburgh University,
Informatics Forum, 10 Crichton Street,
Edinburgh
EH8 9AB United Kingdom
olemon@inf.ed.ac.uk
Phone +44 (131) 650 4443 - Fax +44 (131) 650 4587

Copies of reports and other material can also be accessed via the project's administration homepage, `http://www.classic-project.org`

Contents

Executive summary

This document describes the shared dialogue context model derived from the module interfaces for the CLASSiC Architecture, as defined in the CLASSiC project Deliverable D5.1.1 [1].

Formally, the context model is an XML Schema which describes the dialogue context features required for each CLASSiC SDS component to carry out its processing, as well as the context features supplied by each component. This serves as a reference document for system development in the project, as well as a target data logging format.

1 Context Features Required and Supplied by CLASSIC Components

The CLASSIC architecture is designed to provide a unified statistical model of both the sources of uncertainty (speech recognition, language understanding, language generation) and constraints on uncertainty, thereby allowing multiple possible analyses to be represented, maintained, and reasoned with. The dialogue context must therefore represent both uncertainty about the current state, and the history of the dialogue, in a way which is tractable, and accessible to different Spoken Dialogue System components.

To this end, we provide a definition of the "dialogue context at time t" which is maintained across the CLASSIC processing modules.

This context definition's purpose is to:

- serve as a reference model for SDS components and interfaces developed in the project
- serve as a target logging format for CLASSIC systems.

The proposed CLASSIC architecture is illustrated in figure 1.

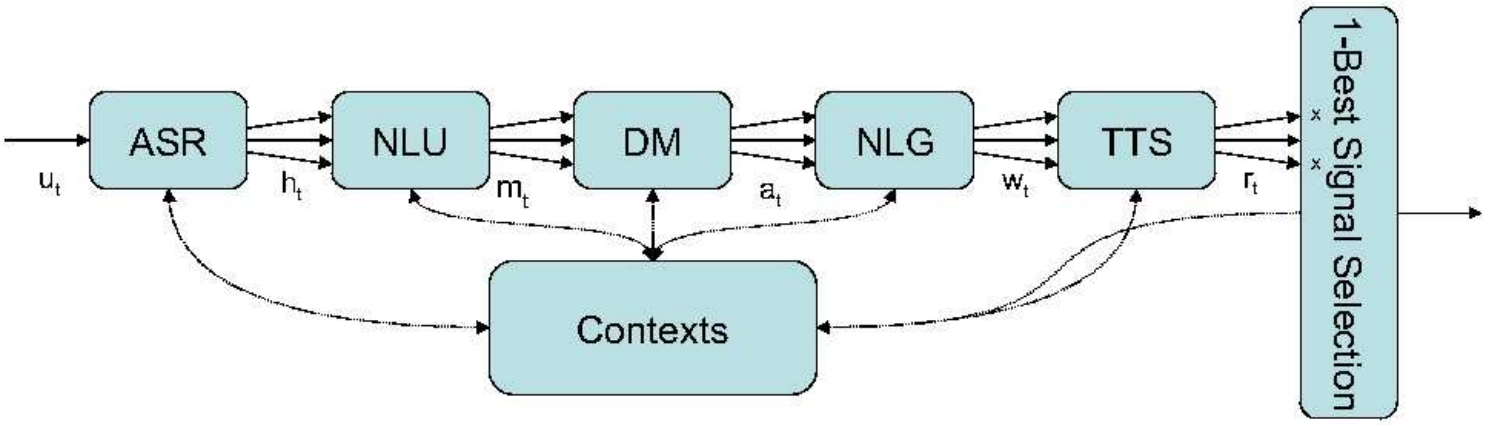

Figure 1: Overview of the CLASSIC Architecture

Figure 2 presents the notation which we use to discuss the mathematical treatment of uncertainty within the CLASSIC architecture.

We now describe the current context features which each module supplies to the dialogue context and which features it requires for its processing. These features are based on the initial CLASSIC architecture as described in Deliverable 5.1.1 [1] and will be extended and refined during the project.

These context features are defined and summarised in the Context Model's Document Type Definition (DTD) and XML Schema, see Appendices.

u_t	speech signal at t
h_t	ASR hypothesis for u_t
m_t	SLU hypothesis for u_t
s_t	state hypothesis after u_t
a_t	system actions after u_t
V_t	history of $u_{t'}$ and $a_{t'}$, $t' \leq t$
$P(s_t\|V_{t-1},u_t)$	belief state distribution

Figure 2: Notation

1.1 Automatic Speech Recognition

The ASR component supplies N speech recognition hypotheses (with confidence scores) to the dialogue context after each user turn.

The ASR component provides $P(h_t|u_t,V_{t-1})$ for each h_t it outputs.

In the initial system the ASR Module does not itself require any context information. However, recent experiments have shown that contextual information from a User Simulation can significantly improve speech recognition performance for ATK in the TownInfo task [2].

1.2 Spoken Language Understanding

The SLU component requires N hypotheses from the ASR component, and it supplies semantic/pragmatic interpretations for each hypothesis (as described in the Cambridge Dialogue Act scheme [1]), along with a confidence score in each interpretation.

The SLU module estimates $\sum_{h_t} P(m_t|h_t,V_{t-1})P(h_t|u_t,V_{t-1})$ for each m_t it outputs.

SLU may choose to ignore the dialogue history entirely, or it may condition on observable context, such as the preceding system prompt. But it cannot choose to condition on information that varies across the DM's different dialogue context states, such as one of the ASR hypotheses from an earlier utterance. To make a clear division between these two types of information, we only allow conditioning on system prompts, and not on any information derived from user utterances. More specifically, we currently envisage that the preceding *system* dialogue act may optionally be used as context. In addition, the *surface text form* might also be used, when available from the NLG/TTS components.

1.3 Dialogue Management

The DM module maintains a representation of dialogue context which includes all information relevant to making dialogue act decisions. Probability distributions over possible dialogue states

are used to represent uncertainty in the dialogue context [3]. The dialogue context at any time therefore includes this distribution over possible dialogue states. The calculation of these probability distributions exploits the probabilistic framework of the CLASSIC architecture.

Note that the dialogue context contains the entire belief state distribution at time t.

The modules which precede the DM module (ASR and SLU) provide the probabilites which the DM module needs to compute its state transition equation.

The state transition equation can be written as follows (see [1]):

$$\overbrace{\sum_{m_t}\sum_{s_{t-1}} \frac{P(m_t|s_{t-1},a_{t-1})P(s_t|s_{t-1},a_{t-1},m_t)P(s_{t-1}|V_{t-2},u_{t-1})}{P(m_t|V_{t-1})Z(V_t)}}^{\text{DM}} \quad \overbrace{\sum_{h_t} P(m_t|h_t,V_{t-1})}^{\text{SLU}} \quad \overbrace{P(h_t|u_t,V_{t-1})}^{\text{ASR}}$$

Figure 3: State Transition Equation, with system components

Here we note that the following terms represent different elements of the dialogue context required by the Dialogue Manager:

- $P(s_{t-1}|V_{t-2},u_{t-1})$ is the previous belief state.
- $P(h_t|u_t,V_{t-1})$ reflects the speech recognizer's confidence in a hypothesis.
- $P(m_t|h_t,V_{t-1})$ reflects multiple possible interpretations of h_t, generated by ambiguity in the SLU module.
- $P(m_t|V_{t-1})$ is the prior over SLU outputs.
- $P(m_t|s_{t-1},a_{t-1})$ is the user model's score of the probability of each possible interpretation m_t, in context.

Note that V_{t-1} includes all the information which can be *unambiguously* extracted from the history of the dialogue prior to u_t. V_{t-1} is used in these equations to designate to what extent different probabilites can make use of context information. The extent to which different modules exploit this context is discussed in the individual modules.

Note that the XML Schema presented later in this document represents the belief state partitions in each context, e.g. that the user is believed to want cheap indian food with probability 0.2, or cheap italian with probability 0.8. While the DIPPER and France Telecom Dialogue Managers used in the project use different internal representations of dialogue state, both can compute distributions over possible user goals, which can be logged using the same mechanism.

1.4 User Model

The User Model computes $P(m_t|s_{t-1}, a_{t-1})$, and therefore requires access to the system action in the preceding turn and the state hypotheses at the preceding turn, as well as the current SLU hypotheses.
It supplies a probability for each of the SLU hypotheses generated by the current user input.

1.5 Natural Language Generation

As discussed in [1] the CLASSIC NLG component requires the following contextual information, for database search applications such as TownInfo:

- Set of items in the database matching user's current constraints, and their attributes/ slot-values
- Filled slots (and the order in which the user has filled them)
- User preference model (dispositional preferences, e.g. price is more important than location, food-type more important than price.)
- Lexical items spoken by the user, in order (from the most probable hypotheses).

Here we note that CLASSIC HIS dialogue manager does not have a notion of filled "slots" but instead a probability distribution over possible user goals. This representation can be used to compute the current most likely user goals, which is conceptually equivalent to the current filled slots. The CLASSIC DIPPER dialogue manager explores an alternative representation, a distribution over states which differ in terms of which values fill different slots, which can be used to compute the same context information [4].

1.6 Text To Speech

No particular context features are required by the TTS modules. However, they will be able to supply a predicted *cost* (a prediction of the synthesis quality) for synthesizing any string. This may be used by future NLG modules as features in output planning.

1.7 The CLASSIC DTD and XML Schema

The Document Type Definition (DTD) and XML Schema supplied in the Appendices are shared between the CLASSIC partners and represent the requirements noted above. They will be refined and extended throughout the project.
In addition to the information noted above, the context representation DTD and Schema also define a number of reward features, which can be used to log the reward associated with particular states and whole dialogues. These features are based on the PARADISE evaluation scheme used in [5] and modified in [6, 7].

The DTD and Schema also contain optional features which may be used in corpus development and analysis, after data collection. For example the *utterance_transcription* feature.

1.8 Moving from TownInfo to SelfHelp

The CLASSiC project works on 2 genres of dialogue: interactive database search applications such as TownInfo and troubleshooting systems such as the SelfHelp system for modem installation.

Most of the context definitions above are general across these 2 genres. The notion of a user "goal" or "slot" is replaced by the world-state information supplied by the user in the SelfHelp domain, for example that their connection light is red or that they have plugged in the broadband cable to the modem.

The NLG component in the SelfHelp application does not "offer" items to the user, but provides them with instructions to follow or questions to answer. In the SelfHelp domain then, "data_base hits" will be empty. The user preference model will also be modified to contain an estimation of the type of user the system is dealing with, for example expert or novice.

2 Example XML Context Representations

We now present a short example of the Context Model, covering the following dialogue turns in the TownInfo domain:

- User turn 2: “Cheap italian food”
- System turn 3: “Ciao Roma is an Italian restaurant , and it has the best overall quality amongst the selected restaurants.”

The following XML document shows 2 contexts. The first (state number 4) is after the interpretation of the above user turn, and the second (state number 5) is after the generation of the following system turn. Dialogue History is encoded by the sequence of states.
In state number 4 there are 2 ASR hypotheses, “cheap indian food” and “cheap italian food”. These lead to 2 SLU hypotheses :

- $inform(food\_type = indian, price = cheap), 0.7$
- $inform(food\_type = italian, price = cheap), 0.5$

These hypotheses are scored by the User Model, and lead to 2 partitions in the belief space, one where the user is believed to want cheap indian food with probability 0.2, and the other cheap italian with probability 0.8.
The $database\_hits$ returned are therefore 2 items, both of which are cheap italian restaurants.
In turn number 5 there is no user input, but the system has the Communicative Goal (from the Dialogue Manager) to “Offer” an item to the user. The NLG component decides to generate this as:
“$recommend([food, quality], Ciao\_Roma)$” and the Realizer component provides the following 2 options:

“Ciao Roma is an Italian restaurant , and it has the best overall quality amongst the selected restaurants” and “Ciao Roma has the best overall quality. It is an Italian Restaurant.”

The TTS component selects the latter option as the one with the lowest cost, which is output by the system in this turn.
In both these contexts, the user preference model states that this user finds the attributes QUALITY, FOOD-TYPE, and PRICE to be most important, in that order.
This XML document conforms to the DTD and Schema given in the Appendices.

```
<?xml version='1.0' encoding='UTF-8'?>

<corpus>

  <context state_number="4">
    <turn_level
                turn_number="4"
                speaker="user"
                utterance_start_time="17.14.2578"
                utterance_end_time="17.16.804"
                utterance_number="2"/>

    <asr>
      <hypothesis n="1" p="0.8">
        cheap indian food
      </hypothesis>
      <hypothesis n="2" p="0.2">
        cheap italian food
      </hypothesis>
      <transcription_input>
        cheap italian food
      </transcription_input>
    </asr>

    <slu>
      <interpretation n="1" p="0.7">
        <dialogue_act type="inform">
          <key_value key="food"  value="indian"/>
          <key_value key="price" value="cheap"/>
        </dialogue_act>
      </interpretation>
      <interpretation n="2" p="0.5">
        <dialogue_act type="inform">
          <key_value key="food"  value="italian"/>
          <key_value key="price" value="cheap"/>
        </dialogue_act>
      </interpretation>

      <sem_transcription_input>
        <dialogue_act type="inform">
          <key_value key="food"  value="italian"/>
          <key_value key="price" value="cheap"/>
```

```
    </dialogue_act>
  </sem_transcription_input>
</slu>

<user_model>
  <interpretation_probability n="1" p="0.6"/>
  <interpretation_probability n="2" p="0.2"/>
</user_model>

<dialogue_manager>
  <partition n="1" p="0.8">
    <key_value key="food"  value="italian"/>
    <key_value key="price" value="cheap"/>
  </partition>
  <partition n="2" p="0.2">
    <key_value key="food"  value="indian"/>
    <key_value key="price" value="cheap"/>
  </partition>
</dialogue_manager>

<database_hits hits="2">
  <item>
    <key_value key="name"     value="Ciao Roma"/>
    <key_value key="food"     value="italian"/>
    <key_value key="price"    value="cheap"/>
    <key_value key="quality"  value="good"/>
    <key_value key="location" value="centre"/>
  </item>
  <item>
    <key_value key="name"     value="Mamma Mia"/>
    <key_value key="food"     value="italian"/>
    <key_value key="price"    value="cheap"/>
    <key_value key="quality"  value="medium"/>
    <key_value key="location" value="centre"/>
  </item>
</database_hits>

<nlg/>

<tts/>

<user_preference_model>
  <preference name="price"/>
```

```
      <preference name="location"/>
      <preference name="food"/>
    </user_preference_model>

    <reward use_system_in_future="0"
            user_ID="user32"
            system_name="0"
            date="12/12/2008"
            dialogue_duration="0"
            start_time="17.13.25"
            end_time="null"
            actual_task_completion="0"
            perceived_task_completion="0"
            task_ease="0"
            comprehension_ease="0"
            user_expertise="0"
            system_behaved_as_expected="0"/>

  </context>

  <context state_number="5">
    <turn_level
            turn_number="5"
            speaker="system"
            utterance_start_time="17.20.347"
            utterance_end_time="17.24.506"
            utterance_number="3"/>

    <asr/>

    <slu/>

    <user_model/>

    <dialogue_manager>
      <partition n="1" p="0.8">
        <key_value key="food"  value="italian"/>
        <key_value key="price" value="cheap"/>
      </partition>
      <partition n="2" p="0.2">
        <key_value key="food"  value="indian"/>
        <key_value key="price" value="cheap"/>
      </partition>
```

```
</dialogue_manager>


<database_hits hits="2">
  <item>
    <key_value key="name"     value="Ciao Roma"/>
    <key_value key="food"     value="italian"/>
    <key_value key="price"    value="cheap"/>
    <key_value key="quality"  value="good"/>
    <key_value key="location" value="centre"/>
  </item>
  <item>
    <key_value key="name"     value="Mamma Mia"/>
    <key_value key="food"     value="italian"/>
    <key_value key="price"    value="cheap"/>
    <key_value key="quality"  value="medium"/>
    <key_value key="location" value="centre"/>
  </item>
</database_hits>

<nlg communicative_goal="Offer">
  <content_plan>
    <dialogue_act type="recommend">
      <key_value key="food"/>
      <key_value key="quality"/>
      <key_value key="name" value="Ciao_Roma"/>
    </dialogue_act>
  </content_plan>
  <realizer_outputs>
    <output>
        Ciao Roma is an Italian restaurant , and it has the best
        overall quality amongst the selected restaurants.
    </output>
    <output>
        Ciao Roma has the best overall quality.
        It is an Italian Restaurant.
    </output>
  </realizer_outputs>
</nlg>

<tts>
  <output cost="0.1">
    Ciao Roma is an Italian restaurant , and it has the best overall quality
```

```
        amongst the selected restaurants.
      </output>
    </tts>

    <user_preference_model>
      <preference name="price"/>
      <preference name="location"/>
      <preference name="food"/>
    </user_preference_model>

    <reward use_system_in_future="0"
            user_ID="user32"
            system_name="0"
            date="12/12/2008"
            dialogue_duration="0"
            start_time="17.13.25"
            end_time="0"
            actual_task_completion="0"
            perceived_task_completion="0"
            task_ease="0"
            comprehension_ease="0"
            user_expertise="0"
            system_behaved_as_expected="0"/>

  </context>

</corpus>
```

References

[1] Ghislain Putois, Steve Young, James Henderson, Oliver Lemon, Verena Rieser, Xingkun Liu, Philippe Bretier, and Romain Laroche. Initial communication architecture and module interface definitions. Deliverable D5.1.1, CLASSIC Project, 2009.

[2] Oliver Lemon and Ioannis Konstas. User Simulations for context-sensitive speech recognition in Spoken Dialogue Systems. In *Proceedings of EACL*, 2009.

[3] M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, K. Yu, and S. Young. Training and evaluation of the HIS POMDP dialogue system in noise. In *Proceedings of SIGDial*, 2008.

[4] James Henderson and Oliver Lemon. Mixture Model POMDPs for Efficient Handling of Uncertainty in Dialogue Management. In *Proceedings of ACL*, 2008.

[5] Marilyn A. Walker, Candace A. Kamm, and Diane J. Litman. Towards Developing General Models of Usability with PARADISE. *Natural Language Engineering*, 6(3), 2000.

[6] James Henderson, Oliver Lemon, and Kallirroi Georgila. Hybrid reinforcement/supervised learning of dialogue policies from fixed datasets. *Computational Linguistics*, 34(4):487–513, 2008.

[7] Oliver Lemon, Kallirroi Georgila, and James Henderson. Evaluating Effectiveness and Portability of Reinforcement Learned Dialogue Strategies with real users: the TALK TownInfo Evaluation. In *IEEE/ACL Spoken Language Technology*, 2006.

A The CLASSiC Context Model DTD

```
<!--
    An example how to use this DTD from your XML document:

    <?xml version="1.0"?>

    <!DOCTYPE corpus SYSTEM "state1.dtd">

    <corpus>
    ...
    </corpus>
-->

<!--- Top Level tag for a whole corpus -->
<!ELEMENT corpus ( context+ ) >

<!--- Entire  context representation at time t -->
<!ELEMENT context ( turn_level, asr, slu, user_model, dialogue_manager, database_hits,
                    nlg, tts, user_preference_model, reward ) >
<!ATTLIST context state_number CDATA #REQUIRED >

<!--- Speech recognition hypotheses and transcription -->
<!ELEMENT asr ( hypothesis*, transcription_input? ) >


<!--- Database items matching current user goals in the top partition,
<!ELEMENT database_hits ( item+ ) >
<!ATTLIST database_hits hits CDATA #REQUIRED >

<!ELEMENT dialogue_act ( key_value+ ) >
<!ATTLIST dialogue_act type CDATA #REQUIRED >

<!--- The Dialogue Manager state: partitions and their probabilities -->
<!ELEMENT dialogue_manager ( partition+ ) >


<!--- ASR and SLU Hypothesis and confidence scores -->
<!ELEMENT hypothesis ( #PCDATA ) >
<!ATTLIST hypothesis n CDATA #REQUIRED >
<!ATTLIST hypothesis p CDATA #REQUIRED >
```

```
<!ELEMENT interpretation ( dialogue_act ) >
<!ATTLIST interpretation n CDATA #REQUIRED >
<!ATTLIST interpretation p CDATA #REQUIRED >

<!ELEMENT interpretation_probability EMPTY >
<!ATTLIST interpretation_probability n CDATA #REQUIRED >
<!ATTLIST interpretation_probability p CDATA #REQUIRED >

<!--- Database Records -->
<!ELEMENT item ( key_value+ ) >

<!--- Key Values elements for communicative acts -->
<!ELEMENT key_value EMPTY >
<!ATTLIST key_value key CDATA #REQUIRED >
<!ATTLIST key_value value CDATA #IMPLIED >

<!--- Natural Language Generator information: Communicative Goal, Content Plan,
and Realizer outputs -->
<!ELEMENT nlg ( content_plan?, realizer_outputs? ) >
<!ATTLIST nlg communicative_goal CDATA #IMPLIED >

<!--- NLG content plan -->
<!ELEMENT content_plan ( dialogue_act ) >

<!--- NLG realizer Outputs -->
<!ELEMENT realizer_outputs ( output+ ) >

<!--- TTS Outputs -->
<!ELEMENT output ( #PCDATA ) >
<!ATTLIST output cost CDATA #IMPLIED >

<!ELEMENT partition ( key_value+ ) >
<!ATTLIST partition n CDATA #REQUIRED >
<!ATTLIST partition p CDATA #REQUIRED >

<!ELEMENT preference EMPTY >
<!ATTLIST preference name CDATA #REQUIRED >



<!--- Reward logging information -->
<!ELEMENT reward EMPTY >
<!ATTLIST reward actual_task_completion CDATA #REQUIRED >
```

```
<!ATTLIST reward comprehension_ease CDATA #REQUIRED >
<!ATTLIST reward date CDATA #REQUIRED >
<!ATTLIST reward dialogue_duration CDATA #REQUIRED >
<!ATTLIST reward end_time CDATA #REQUIRED >
<!ATTLIST reward perceived_task_completion CDATA #REQUIRED >
<!ATTLIST reward start_time CDATA #REQUIRED >
<!ATTLIST reward system_behaved_as_expected CDATA #REQUIRED >
<!ATTLIST reward system_name CDATA #REQUIRED >
<!ATTLIST reward task_ease CDATA #REQUIRED >
<!ATTLIST reward use_system_in_future CDATA #REQUIRED >
<!ATTLIST reward user_ID CDATA #REQUIRED >
<!ATTLIST reward user_expertise CDATA #REQUIRED >

<!--- Semantics of the transcribed user input -->
<!ELEMENT sem_transcription_input ( dialogue_act ) >

<!--- Spoken Language Understanding interpretations, scores etc -->
<!ELEMENT slu ( interpretation*, sem_transcription_input? ) >

<!--- Transcribed user input -->
<!ELEMENT transcription_input ( #PCDATA ) >

<!--- Speech synthesiser: chosen output string and its cost -->
<!ELEMENT tts ( output? ) >

<!--- Turn-level information for the current utterance -->
<!ELEMENT turn_level EMPTY >
<!ATTLIST turn_level speaker CDATA #REQUIRED >
<!ATTLIST turn_level turn_number CDATA #REQUIRED >
<!ATTLIST turn_level utterance_end_time CDATA #REQUIRED >
<!ATTLIST turn_level utterance_number CDATA #REQUIRED >
<!ATTLIST turn_level utterance_start_time CDATA #REQUIRED >

<!--- User model probabilities for the SLU hypotheses, in
this context -->
<!ELEMENT user_model ( interpretation_probability* ) >

<!--- List of user dispositional preferences, most important attribute first -->
<!ELEMENT user_preference_model ( preference+ ) >
```

B The CLASSiC Context Model XML Schema

```
<?xml version="1.0" encoding="UTF-8" ?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="asr">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="hypothesis" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="transcription_input" minOccurs="0" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="content_plan">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="dialogue_act" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="context">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="turn_level" />
        <xs:element ref="asr" />
        <xs:element ref="slu" />
        <xs:element ref="user_model" />
        <xs:element ref="dialogue_manager" />
        <xs:element ref="database_hits" />
        <xs:element ref="nlg" />
        <xs:element ref="tts" />
        <xs:element ref="user_preference_model" />
        <xs:element ref="reward" />
      </xs:sequence>
      <xs:attribute name="state_number" type="xs:CDATA" use="required" />
    </xs:complexType>
  </xs:element>

  <xs:element name="corpus">
    <xs:complexType>
```

```
    <xs:sequence>
      <xs:element ref="context" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="database_hits">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="item" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="hits" type="xs:CDATA" use="required" />
  </xs:complexType>
</xs:element>

<xs:element name="dialogue_act">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="key_value" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="type" type="xs:CDATA" use="required" />
  </xs:complexType>
</xs:element>

<xs:element name="dialogue_manager">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="partition" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="hypothesis">
  <xs:complexType mixed="true">
    <xs:attribute name="p" type="xs:CDATA" use="required" />
    <xs:attribute name="n" type="xs:CDATA" use="required" />
  </xs:complexType>
</xs:element>

<xs:element name="interpretation">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="dialogue_act" />
```

```
    </xs:sequence>
    <xs:attribute name="p" type="xs:CDATA" use="required" />
    <xs:attribute name="n" type="xs:CDATA" use="required" />
  </xs:complexType>
</xs:element>

<xs:element name="interpretation_probability">
  <xs:complexType>
    <xs:attribute name="p" type="xs:CDATA" use="required" />
    <xs:attribute name="n" type="xs:CDATA" use="required" />
  </xs:complexType>
</xs:element>

<xs:element name="item">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="key_value" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="key_value">
  <xs:complexType>
    <xs:attribute name="key" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:CDATA">
          <xs:enumeration value="food" />
          <xs:enumeration value="location" />
          <xs:enumeration value="name" />
          <xs:enumeration value="price" />
          <xs:enumeration value="quality" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="value" type="xs:string" use="optional" />
  </xs:complexType>
</xs:element>

<xs:element name="nlg">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="content_plan" minOccurs="0" />
      <xs:element ref="realizer_outputs" minOccurs="0" />
```

```
    </xs:sequence>
    <xs:attribute name="communicative_goal" type="xs:CDATA" use="optional" />
  </xs:complexType>
</xs:element>

<xs:element name="output">
  <xs:complexType mixed="true">
    <xs:attribute name="cost" type="xs:CDATA" use="optional" />
  </xs:complexType>
</xs:element>

<xs:element name="partition">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="key_value" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="p" type="xs:CDATA" use="required" />
    <xs:attribute name="n" type="xs:CDATA" use="required" />
  </xs:complexType>
</xs:element>

<xs:element name="preference">
  <xs:complexType>
    <xs:attribute name="name" type="xs:CDATA" use="required" />
  </xs:complexType>
</xs:element>

<xs:element name="realizer_outputs">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="output" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="reward">
  <xs:complexType>
    <xs:attribute name="end_time" type="xs:CDATA" use="required" />
    <xs:attribute name="task_ease" type="xs:CDATA" use="required" />
    <xs:attribute name="dialogue_duration" type="xs:CDATA" use="required" />
    <xs:attribute name="date" type="xs:string" use="required" />
    <xs:attribute name="user_ID" type="xs:CDATA" use="required" />
    <xs:attribute name="actual_task_completion" type="xs:CDATA" use="required" />
```

```
    <xs:attribute name="start_time" type="xs:CDATA" use="required" />
    <xs:attribute name="system_name" type="xs:CDATA" use="required" />
    <xs:attribute name="comprehension_ease" type="xs:CDATA" use="required" />
    <xs:attribute name="use_system_in_future" type="xs:CDATA" use="required" />
    <xs:attribute name="system_behaved_as_expected" type="xs:CDATA" use="required" /
    <xs:attribute name="perceived_task_completion" type="xs:CDATA" use="required" />
    <xs:attribute name="user_expertise" type="xs:CDATA" use="required" />
  </xs:complexType>
</xs:element>

<xs:element name="sem_transcription_input">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="dialogue_act" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="slu">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="interpretation" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="sem_transcription_input" minOccurs="0" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="transcription_input">
  <xs:complexType mixed="true" />
</xs:element>

<xs:element name="tts">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="output" minOccurs="0" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="turn_level">
  <xs:complexType>
    <xs:attribute name="utterance_end_time" type="xs:CDATA" use="required" />
    <xs:attribute name="speaker" type="xs:CDATA" use="required" />
```

```
      <xs:attribute name="utterance_number" type="xs:CDATA" use="required" />
      <xs:attribute name="utterance_start_time" type="xs:CDATA" use="required" />
      <xs:attribute name="turn_number" type="xs:CDATA" use="required" />
    </xs:complexType>
  </xs:element>

  <xs:element name="user_model">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="interpretation_probability" minOccurs="0" maxOccurs="unbounde
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="user_preference_model">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="preference" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```