Episodic Memory for Companion Dialogue

Gregor Sieber & Brigitte Krenn
Austrian Research Institute for Artificial Intelligence (OFAI)
Interaction Technologies Group
{gregor.sieber,brigitte.krenn}@ofai.at

Outline

- Motivation - why do companions need episodic memory?
- What data should be remembered, and how can it be stored?
- What retrieval mechanisms are required?
- How can memories be used for dialogue?
- Open issues and future work

Motivation

Companions...

- engage in long-term interaction
- may be active while the user is offline
- should learn from their experiences
- should know interests of the user
- should understand references to previous dialogue
- connect internal representations of language with their experiences

Motivation

Companions...

- engage in long-term interaction
- may be active while the user is offline
- should learn from their experiences
- should know interests of the user
- should understand references to previous dialogue
- connect internal representations of language with their experiences

=> dialogue that takes into account what has happened in the past and is able to refer to previous events

Episodic Memory

- Episodic memory (EM):
  - memory of personal events
  - “mental time-travel”
  - spatio-temporal location, actors, feelings, ...
  - reconstructive
- Some applications of EM:
  - reporting on past events [Dias 2007] by summarizing emotionally important memories
  - background stories for non-player characters that affect their actions [Brom 2007]
  - personalization and handling of sensitive information in storytelling companions [Lim et al. 2009]
  - improving performance of robots or agents:
    - artificial life agents [Nuxoll 2007] in game environments
    - learning action strategies for a robot [Dodd and Gutierrez 2005]

Scenario

Our scenario

- companion that answers questions on popular music and gossip about artists, suggests similar music [Krenn et al. 2009]
- user preferences from a web interface
- ontology with popular music data

Companion input

- input class label
- time stamp
- surface string
- RDF triples describing analysis in terms of domain data

  "When was Billy Joel born?" → `data:artist.341 data:birthDate sys:x.`

Output Generation

- from output labels using templates with slots for contextual data
- from RDF representations of domain data

Adding EM to the Scenario: Wish List

- learn selection of actions or tools from past experiences and evaluations
- refer to similar past events
- generate output based on episodic memories and their relation to the current situation
- use memories as a basis for understanding input (anaphora/ elliptic expressions)
- integrate interests of user into dialogue

Episode Encoding

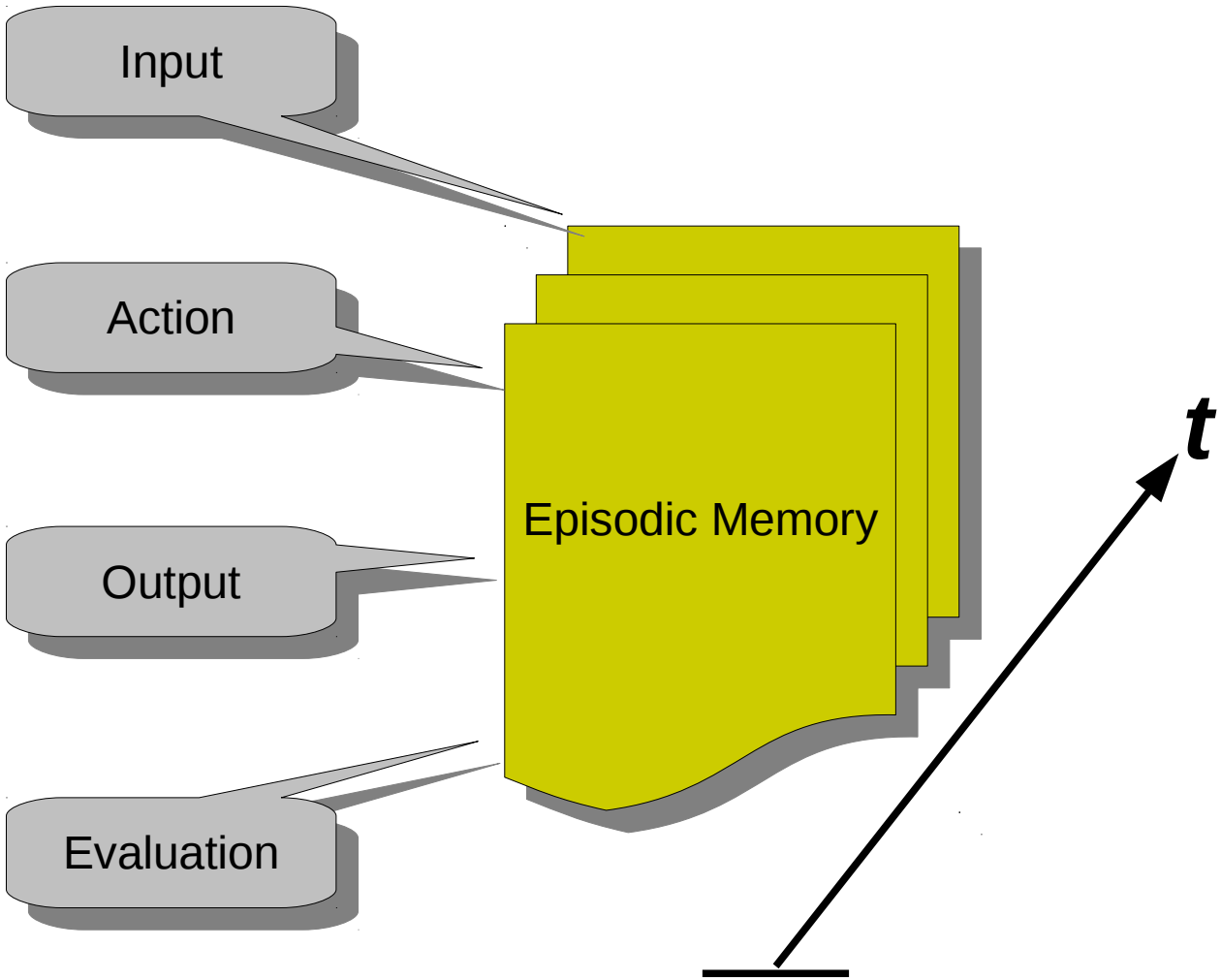
How to encode the data from our scenario?

- possible methods
  - use surface strings
  - transform RDF to keywords
  - serialize RDF data to relational database
- but: information encoded in RDF description and its relation to domain ontology should not be lost!

Proposal:

- describe memories using RDF
- directly store representations of input and output triples
- requires retrieval mechanisms that do not operate on time, moods, or tags but on similarity of individuals and relations

Memory and Episodes



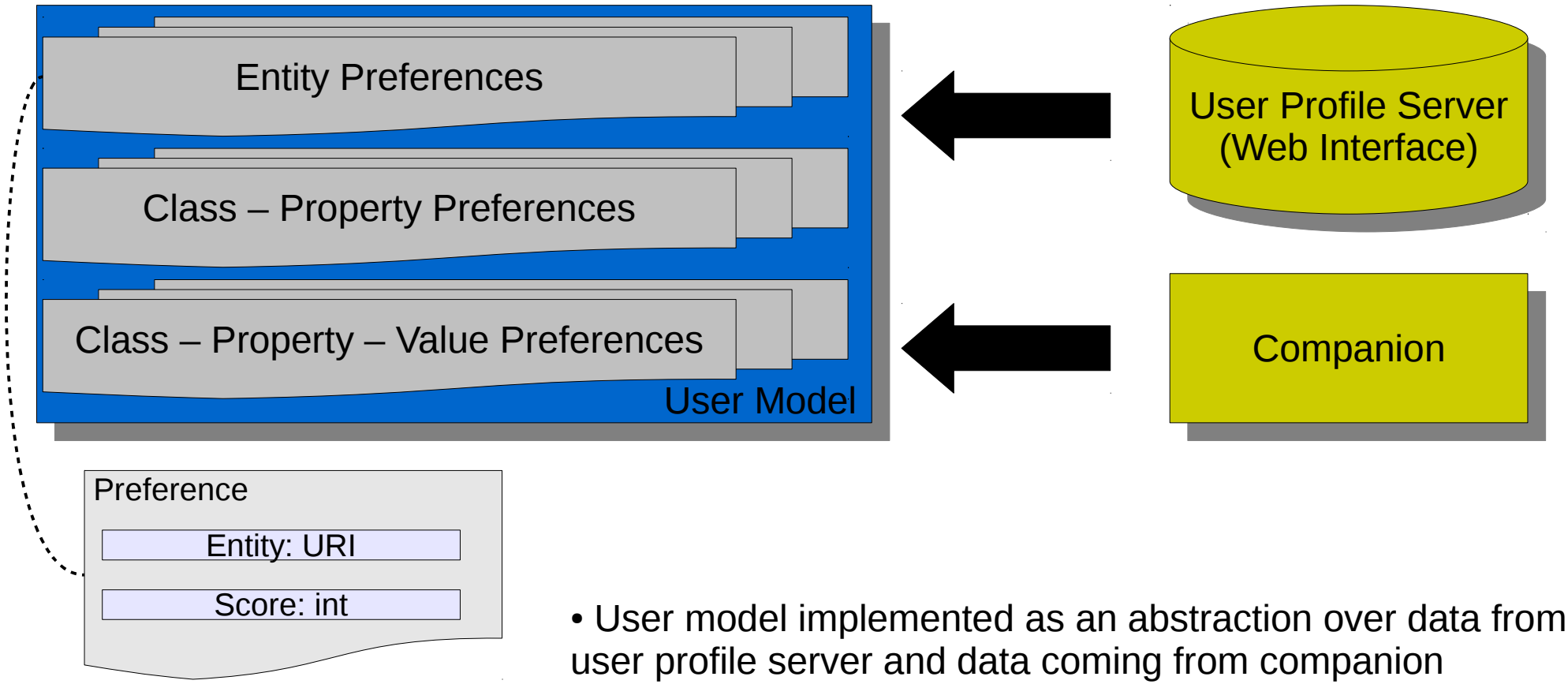
Actions:
- Use question answering
- Find similar episode
- Use pattern match result
- Resolve reference
- Send output

Episodes encode:
- Actors
- Time
- Episode ID
- Sets of RDF triples that encode data from ontology
- Action types
- Evaluation results

Time:
- Relative time category
- Internally: epoch time

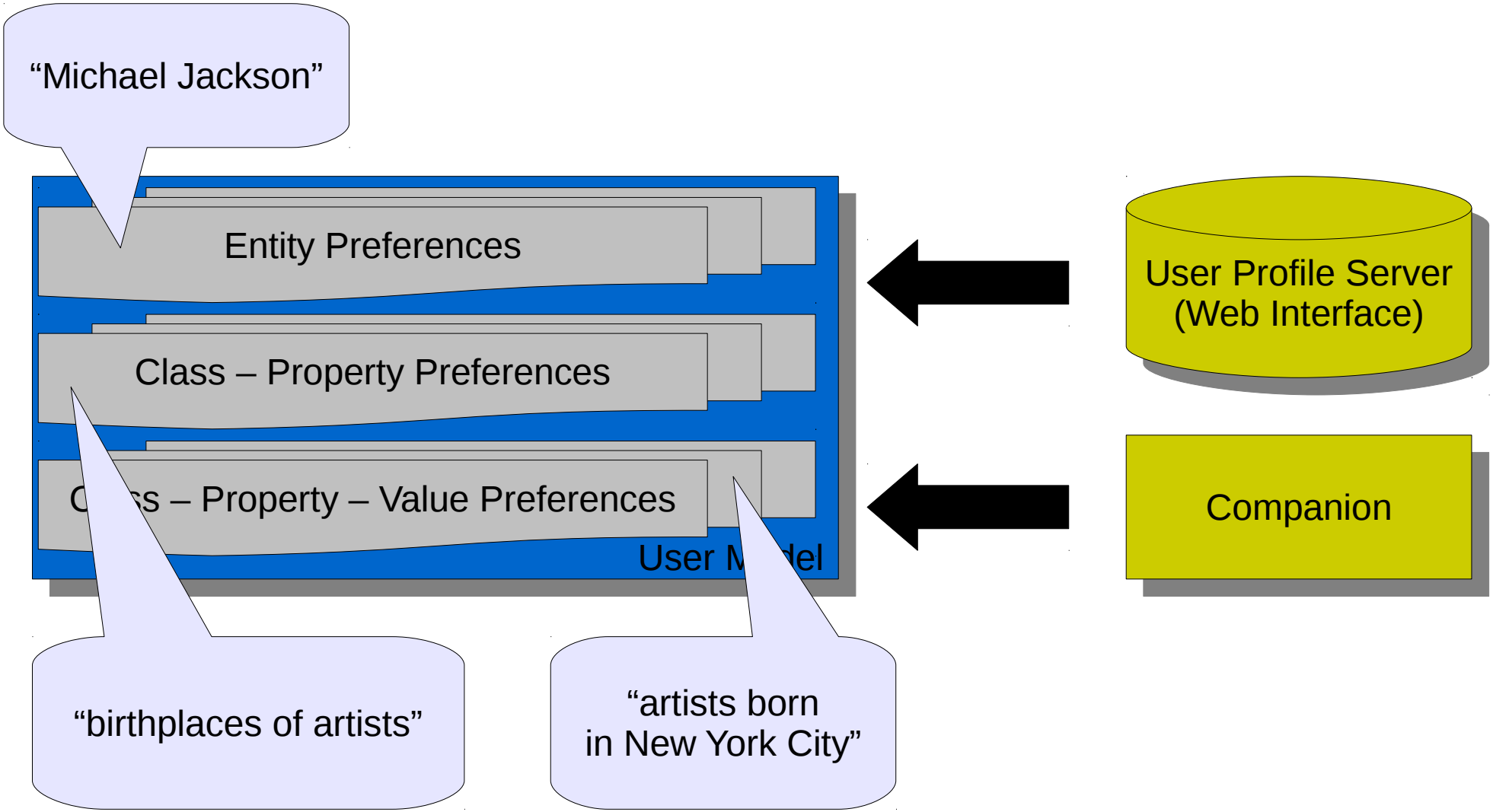
RDF Storage: Sesame (www.openrdf.org)
Elmo used for object persistence

User Model



- User model implemented as an abstraction over data from user profile server and data coming from companion
- Three general types of preferences
- Score is used to rank preferences and to apply decay

User Model


“Michael Jackson”
Entity Preferences
Class – Property Preferences
Class – Property – Value Preferences
User Model
User Profile Server
(Web Interface)
Companion
“birthplaces of artists”
“artists born
in New York City”

Retrieval Mechanisms

- Retrieval methods suggested in related work do not directly apply
  - search by keyword or time alone is not enough
  - vector similarity search would not use RDF structure of data
- Approach:
  - use RDF encoding of companion data
  - retrieve episodes using an RDF query language
  - 3 different retrieval methods
    - finding episodes based on similarity of entities, relations, concepts
    - finding patterns expressed as SeRQL queries in the dialogue manager
    - providing candidates for retrieval of parts of previous utterances necessary to understand current input
  - use of class and domain/range information of properties for retrieval

Search by Similarity

- knowing about similar episodes allows companion to
  - refer to / comment on relation between current and past episodes
  - repeat successful strategies, avoid unsuccessful ones
- search: generate queries from input episode to find
  - identical episodes
  - overlapping entities, properties, classes
  - sorted by nr. of overlapping elements and time
  - depending on application, different rankings may be necessary
- retrieval of evaluations
  - search forward in time from target episode
  - stop when evaluation or next user input is reached

Referring to Previous Dialogue

- user should be able to refer to previous parts of dialogue
- reduce amount of additional external storage mechanisms needed for resolving references
- Examples:

A: “When was Charlie Parker born?”
B: “Charlie Parker was born on August 29th, 1920”.
A: “And what about John Scofield?”

A: “Does Smith own the Porsche?”
B: “Yes, he does.”
A: “And the Mercedes?”

A: “Does Smith own the Porsche?”
B: “Yes, he does.”
A: “And how about the house?”

- not enough information to answer request
- missing information can be retrieved from previous dialogue

Referring to Previous Dialogue

- use information contained in RDF data schema
- search back through context to retrieve missing property from
  - other statements that contain class
  - alternatively, search for properties whose range includes the class
- search context limited by
  - time
  - number of episodes
- simple approach only works for situations where only one candidate is found, otherwise additional selection mechanisms are required

Using Patterns in Memory

- third retrieval method: search for patterns that represent situations that trigger actions or outputs
  - across multiple episodes
  - connecting user model, episodic memory and knowledge base

  Examples:

  - input previously encountered
    - comment and refer back to previous answer, depending on how far back the episode was
  - last input contained property that is among user's interests
    - automatically provide information in this interaction
    - comment on the fact that this is among user interests
    - ask user whether she wants this type of information on a different preferred entity

Using Patterns in Memory

- strong tendency towards one value of a property in the past interactions
  - search for similar cases and communicate result
  - store information in user model

    “You seem to be very interested in artists from New York. Shall I store that as a preference?”
- discover structural similarities between recent topics by combining EM and domain knowledge
  - similarities between recent individuals

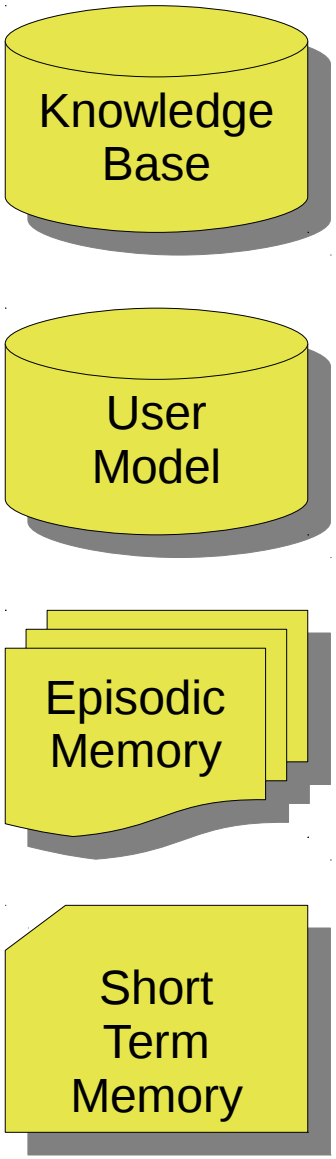
    “Did you know that X,Y,Z share the same home town?”
  - find recently discussed individuals that share property currently under discussion

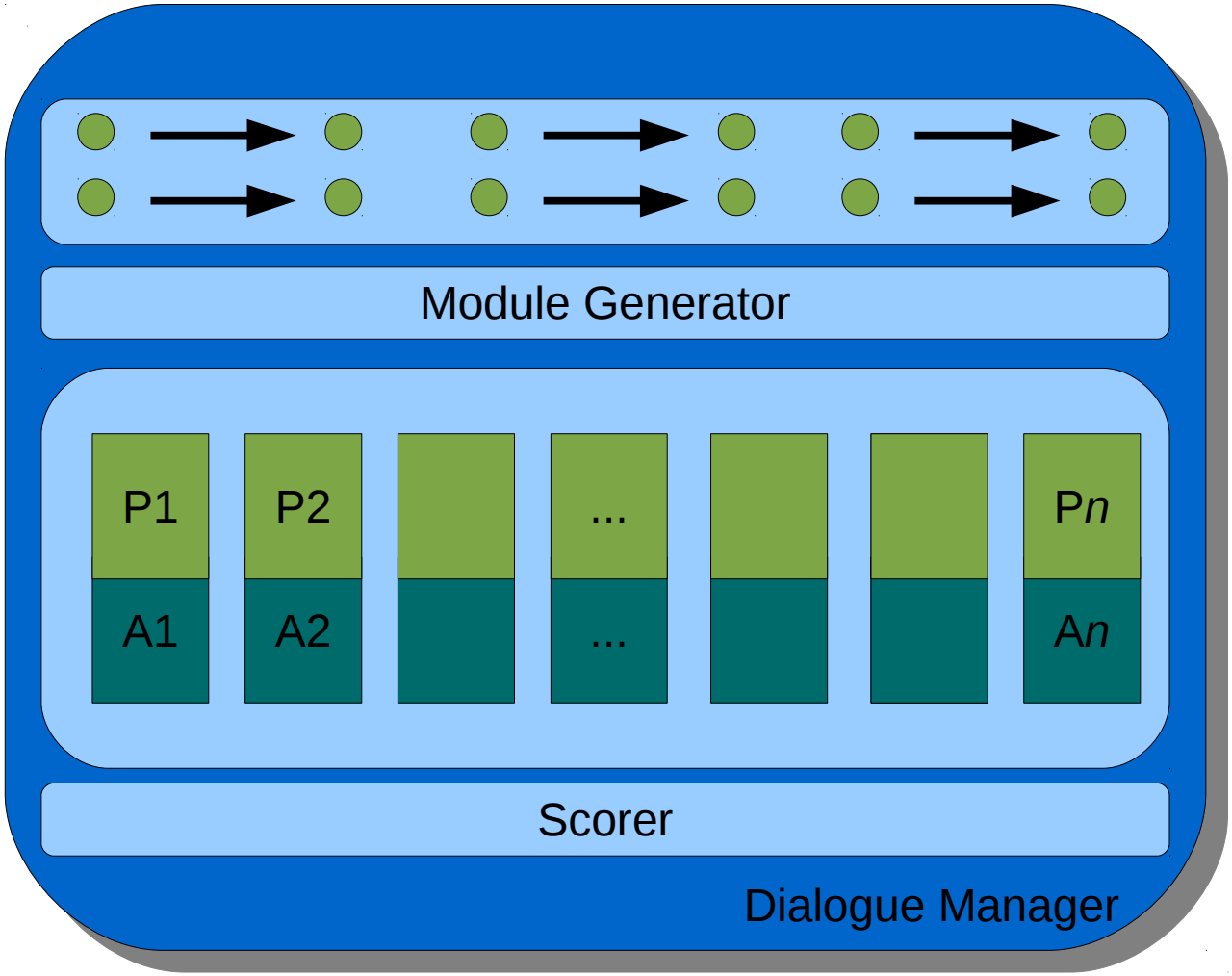
    “By the way, John Frusciante was born in New York, too.”

Dialogue Management

- hybrid approach: rules and patterns with scoring heuristic
- only very basic planning; scheduling of behaviors that depend on user feedback
- modules encode
  - different patterns in memory
  - actions to execute
  - output labels that map to a corresponding template
- winning module is selected by scoring heuristic that takes into account
  - when the module was last used
  - evaluations of previous applications of module
  - preference score of involved domain data

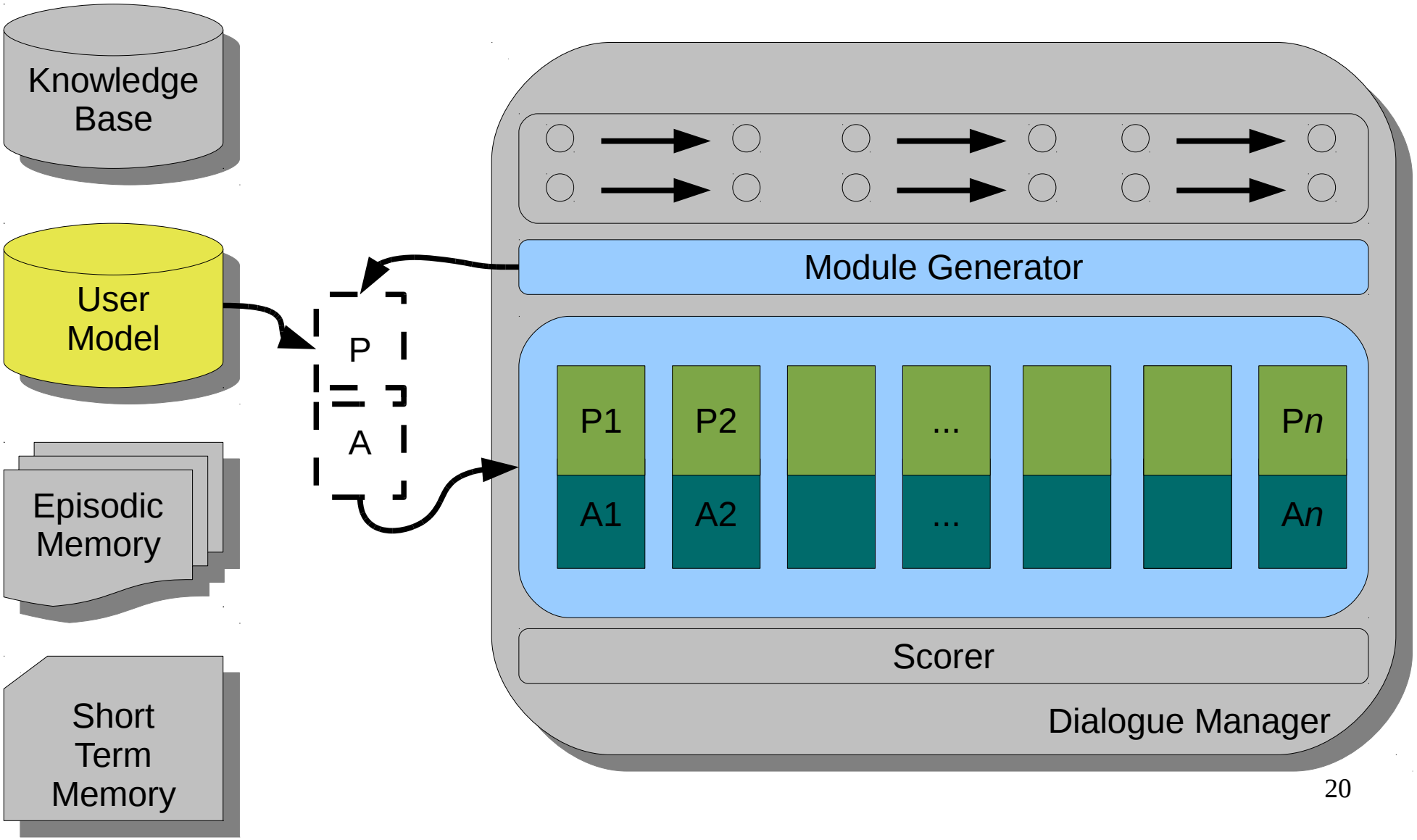
Dialogue Management





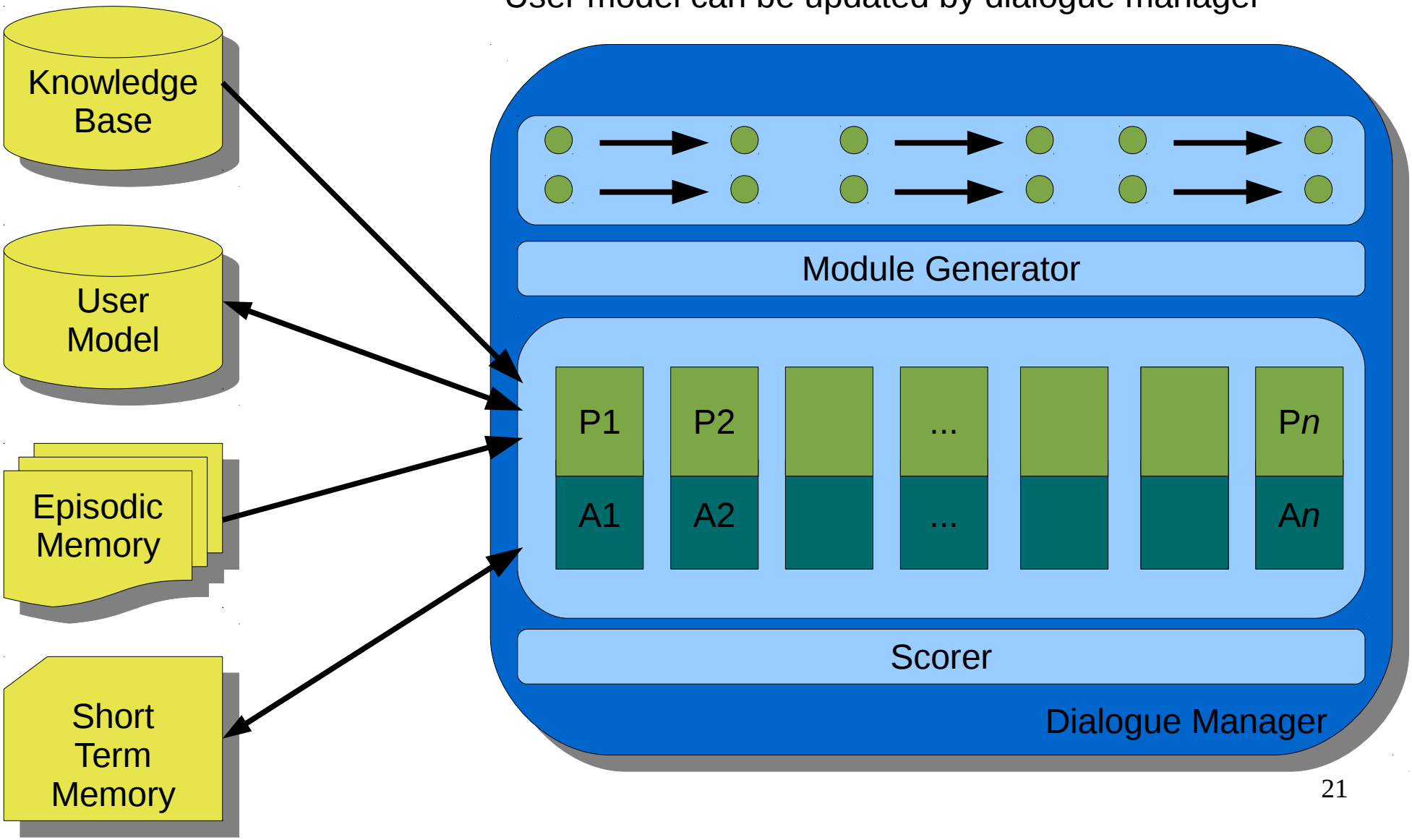
Dialogue Management

- Modules generated by instantiation of templates
- Parameters filled with information from user model



Dialogue Management

- Dialogue manager uses information from EM, STM, knowledge base and user model
- User model can be updated by dialogue manager



Dialogue Management

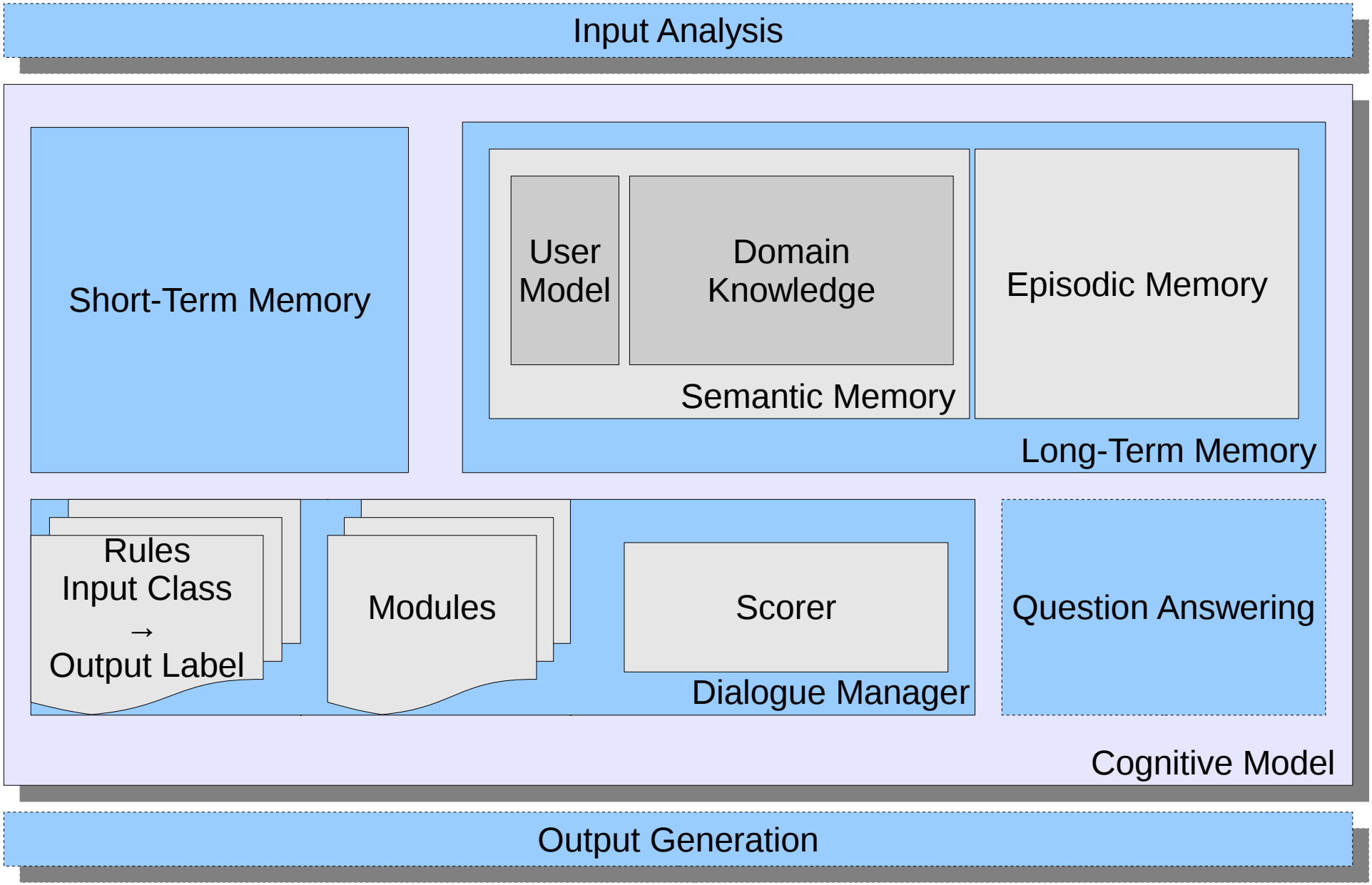
Scoring heuristic:

- assign base score at generation time
- discard modules with no result
- retrieve last episodes from memory and
  - apply *recency penalty* for each occurrence of module
  - apply *negative evaluation penalty* if a negatively evaluated episode exists with the same module and
    - identical episode content
    - entity match
    - class and property match

Conclusion: Results

- we have implemented an RDF-based episodic memory encoding semantic representations of input and output
- we have shown three different methods of retrieving memories using RDF queries
- we have proposed a mechanism for generating output based on patterns in episodic memory
- our data representation makes the different parts of memories interoperable and allows a connection of domain data with experiences of the companion

Conclusion: Companion Architecture



Conclusion: Results

Episodic memories enable the companion to:

- comment on previous interactions
- retrieve past situations and their evaluation
- retrieve parts of previous utterances necessary to understand input
- update user model, discover new interests
- provide additional information tailored to interests of the user

Conclusion: Open Issues

Open Issues

- retrieval speed needs to be optimized
- storage / retrieval efficiency with growing memory
  - growing number of episode → less efficient retrieval
  - current solution: deletion of episodes with oldest time of last retrieval
- evaluation

Possible improvements:

- investigate possible optimizations of queries and indexing
- introduce blending of similar events into one episode
- include a model of emotion as an additional cue to importance of a memory

Thank You.
Questions or Comments?