Modeling Dialogue

Building Highly Responsive Conversational Agents

David Schlangen, Stefan Kopp
with Sören Klett
CITEC // Bielefeld University

Universität Bielefeld

CITEC
Cognitive Interaction Technology
Center of Excellence
Bielefeld University

Overview of Course

- Day 1: Motivation, Phenomena
- Day 2: Technical Challenges, Approaches
- Day 3: Introduction to Technical Framework
- Day 4: Hands-On Project
- Day 5: Reports, Discussion

Takeaways from Day 2

- Dialogue Processing Flow: ASR —NLU — DM — NLG / NVBG — Realizer
  - all components must run incrementally and interact via local updates
- IU model:
  - IS updated with minimal units of information, as soon as hypothesised
  - “higher-level” hypotheses formed on basis of “lower-level” ones
  - IS may have to be revised, in light of newer information
- Hybrid systems: *main* DM plus reactive layer
- Incremental generation and realization allows for reducing latency and adapting more naturally to disturbances

Modeling Dialogue

Building Highly Responsive Conversational Agents

Day 3: Introduction to Technical Framework

David Schlangen, Stefan Kopp
with Sören Klett
CITEC // Bielefeld University

Universität Bielefeld

CITEC
Cognitive Interaction Technology
Center of Excellence
Bielefeld University

Overview of Day 3

- Introducing the framework and demo of a running system
- A more detailed look at the framework
- Setting up technically
- First hands-on experiences

Highly Responsive Agent Framework (HiRAF)

- „simple but robust“ architecture for realizing highly responsive embodied conversational agents
- providing all modules for an end-to-end system
- low-latency, incremental processing
- easily customizable and extensible
- builds on standard components (where available)

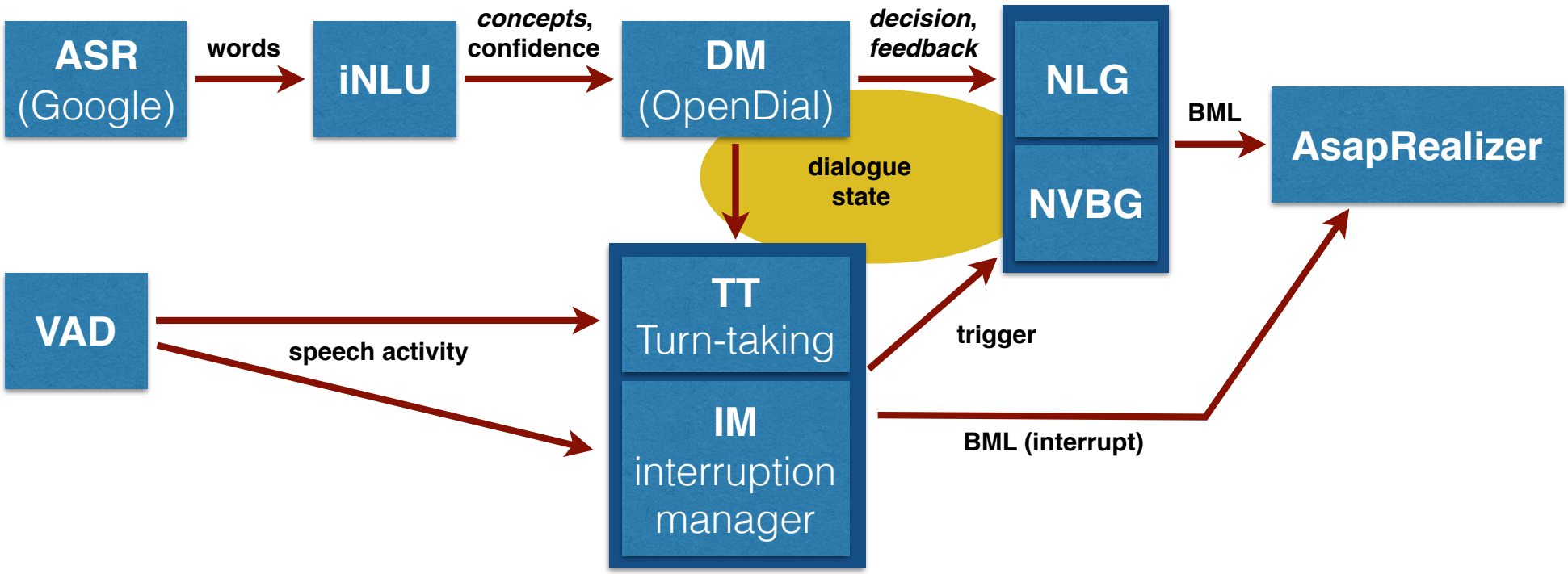
in collab' with Casey Kennington, special thanks to Herwin van Welbergen, Ramin Yaghoubzadeh, Timo Baumann, Pierre Lison

Live demo

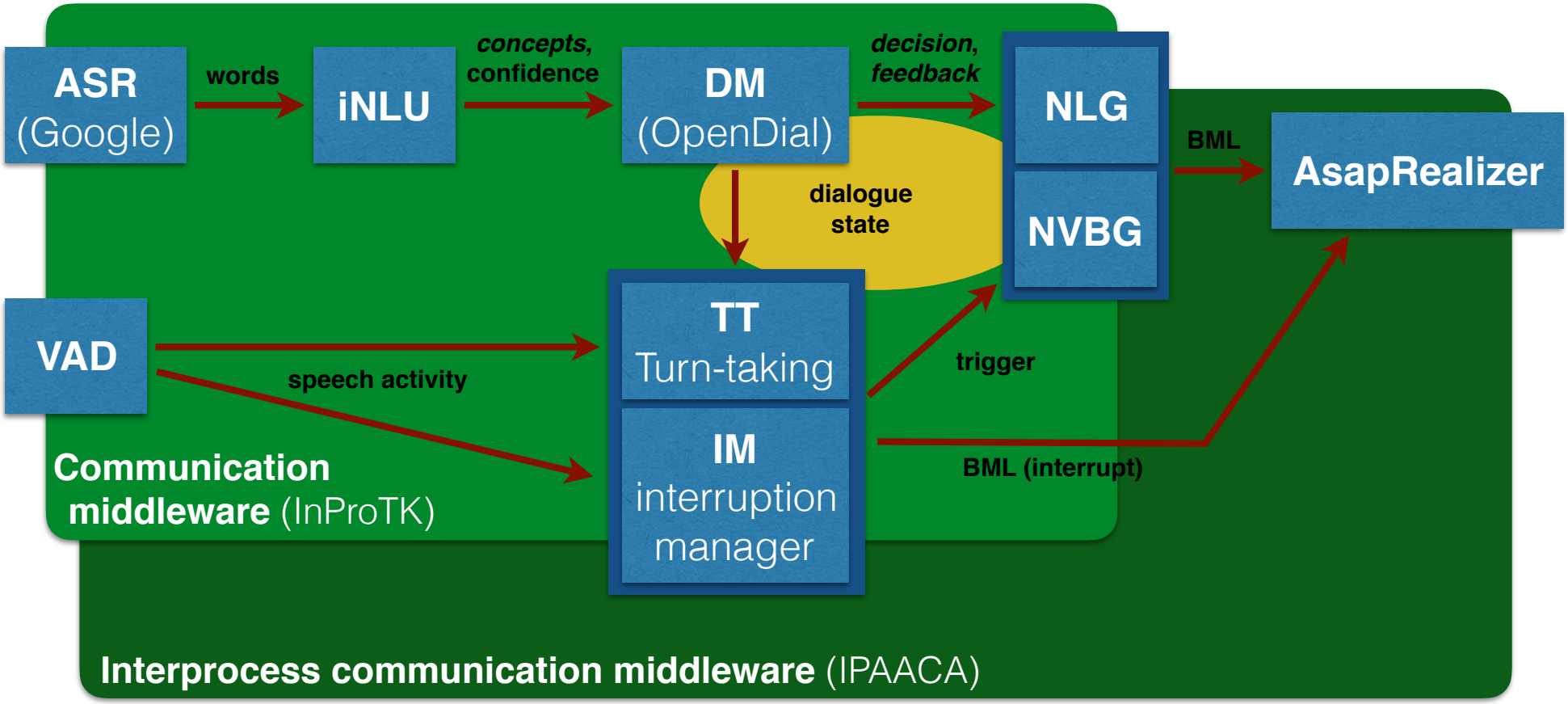
A look at a responsive conversational agent built with the HiRAF framework

Domain:
Quiz bowl (pyramidal tossup)

Overview of modules

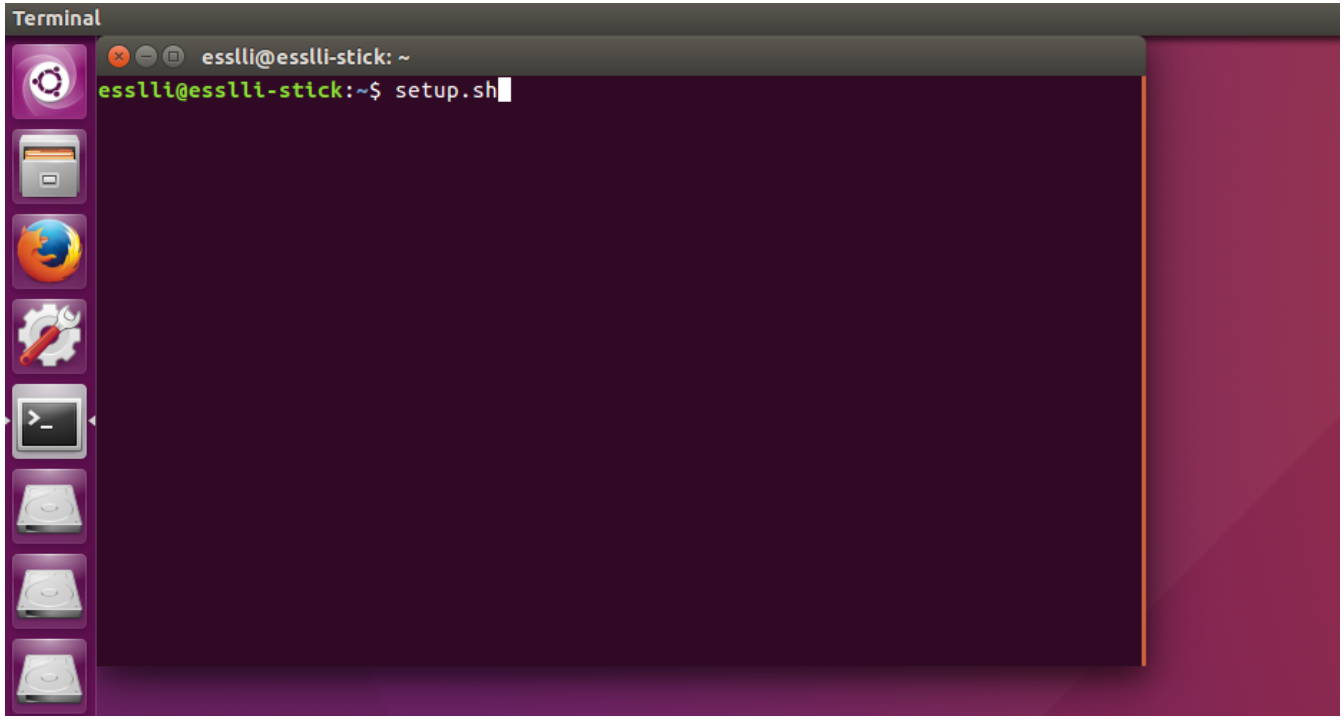


Overview of modules



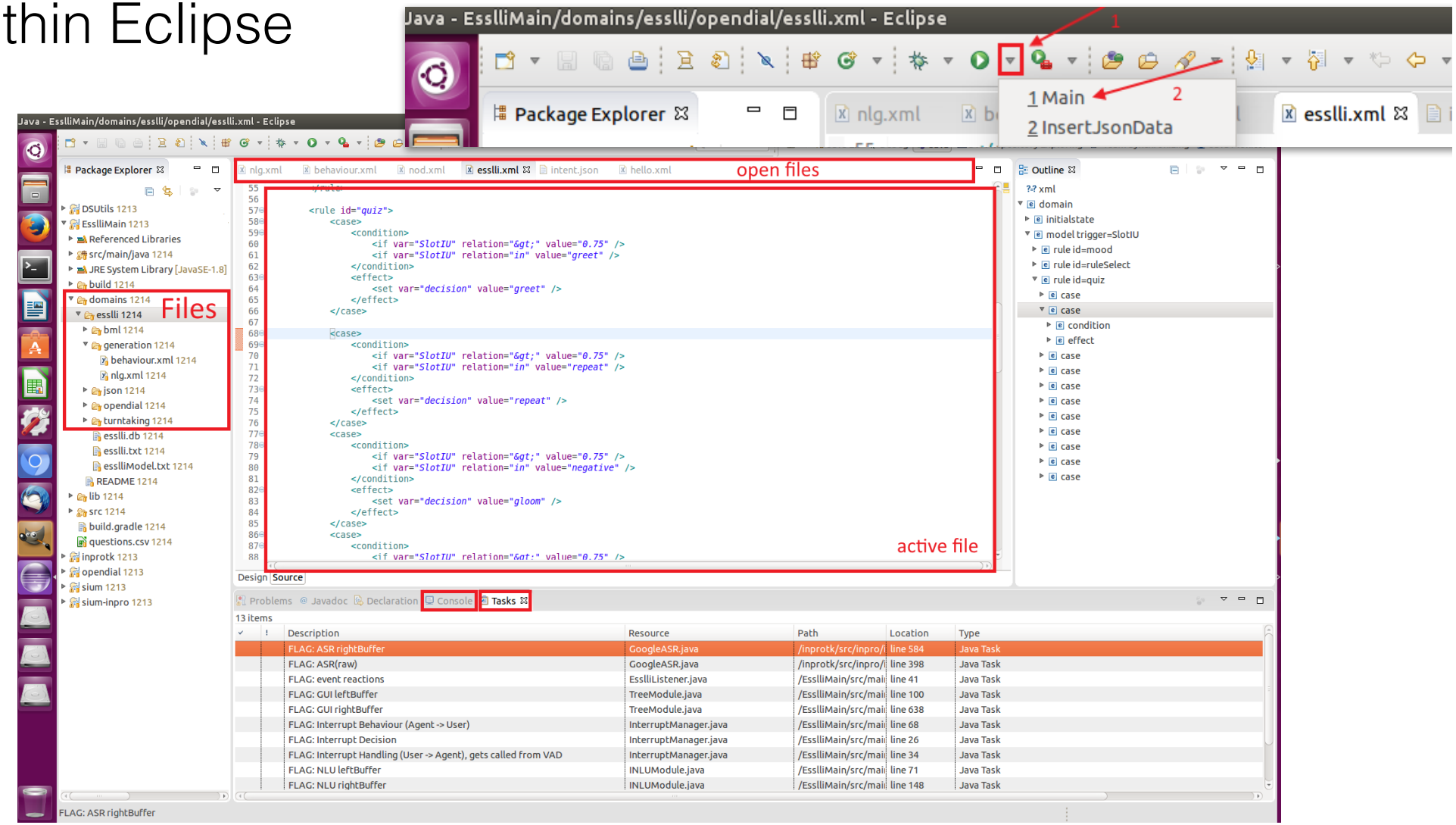
Starting up

1. Boot from the provided USB 3.0-stick (on Mac: hold ALT on startup)
2. Run **`setup.sh`** in a terminal:



Starting up

System starts within Eclipse



Starting up

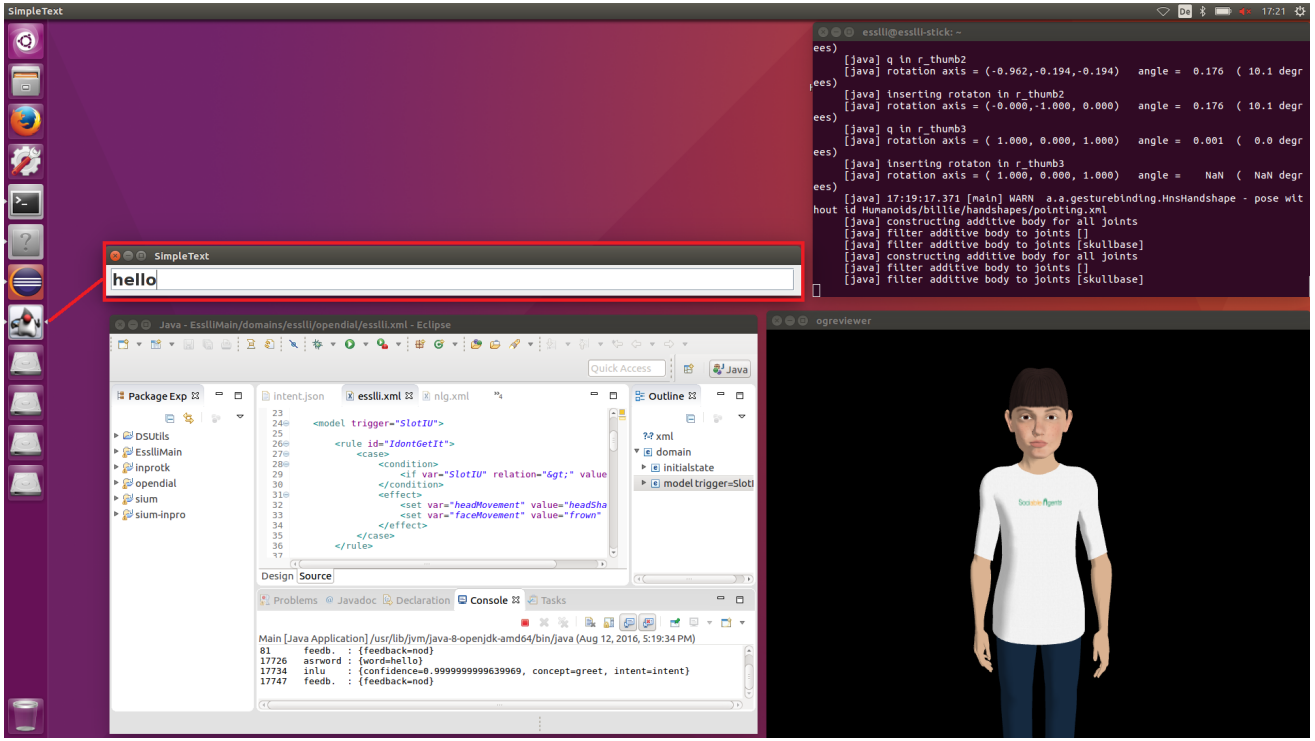
„Billie“ at first

Preparing

Ready

How to interact

Text input
Caution: typing into this field for the first time might have some delay. Make sure the window is on focus (indicated by blinking cursor)



Speech
microphone using GoogleASR (audio gets recognized via google servers, so you need internet connection)

Modules: ASR

slliMain/src/main/java/app/Main.java - Eclipse

Package Explorer

- DSUtils 1213
- > EsslliMain 1213
  - Referenced Libraries
  - > src/main/java 1214
    - > app 1214
      - ConsoleInterface.java 1214
      - > Main.java 1214
      - SigDial1NonIncremental.java
      - SigDial2Incremental.java 121
      - SigDial3Adaptive.java 1214
    - asap.ipaacattsbinding 1214
    - asap.ipaacattsbinding.loader
    - asap.tts.ipaaca 1214
    - config 1214
    - esslli 1214
    - esslli.model 1214
    - jetty 1214
    - model 1214
    - model.db 1214
    - model.functions 1214
    - model.iu 1214
    - module 1214
    - module.opendial 1214
    - sandbox 1214
    - util 1214
  - JRE System Library [JavaSE-1.8]
  - build 1214
  - domains 1214
  - lib 1214
  - > src 1214
  - build.gradle 1214
  - questions.csv 1214
- inprotk 1213
- opendial 1213
- sium 1213
- sium-inpro 1213

nlg.xml | behaviour.xml | nod.xml | esslli.xml | intent.json | hello.xml | EsslliListener.java | Main.java

```
1  package app;
2
3  import inpro.apps.SimpleReco;
39
40 public class Main {
41
42     private static boolean asrActive = false;
43
44     static Logger log = Logger.getLogger("root");
45     public static OutputBuffer outBuffer = new OutputBuffer("main");
46
47     public static boolean openBrowser = false;
48
49     @S4Component(type = DiaTreeSocket.class)
50     public final static String DIATREE_SOCKET = "diatree";
51
52     @S4Component(type = TextBasedFloorTracker.class)
53     public final static String PROP_FLOOR_MANAGER = "textBasedFloorTracker";
54
55     GoogleASR webSpeech;
56     private static PropertySheet ps;
57
58     private static IUDocument iuDocument;
59
60     private static String googleApiKey = "";
61     // private List<PushBuffer> hypListeners;
62     List<EditMessage<IU>> edits = new ArrayList<EditMessage<IU>>();
63
64     private void run() throws InterruptedException, PropertyException, IOException,
65             UnsupportedAudioFileException {
66
67         log.setLevel(Level.OFF);
68
69         Initializer.initializeIpaacaRsb();
70
```

Outline

- app
- > Main 1214
  - asrActive : boolean
  - log : Logger
  - outBuffer : OutputBuffer
  - openBrowser : boolean
  - DIATREE_SOCKET : String
  - PROP_FLOOR_MANAGER : String
  - webSpeech : GoogleASR
  - ps : PropertySheet
  - iuDocument : IUDocument
  - googleApiKey : String
  - edits : List<EditMessage<IU>>
  - run() : void
  - startGoogleASR(ConfigurationManager, RecoComm
    - new Thread() {...}
  - main(String[]) : void
  - notifyListeners(List<PushBuffer>) : void

Problems | Javadoc | Declaration | Console | Tasks

<terminated> Main [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (12.08.2016 10:54:07)

app.Main.java - EsslliMain/src/main/java

Modules: ASR

- let’s have a look at what ASR provides us with

Modules: iNLU

sslliMain/domains/esslli/json/intent.json - Eclipse

Package Explorer

- DSUtils 1213
- > EsslliMain 1213
  - Referenced Libraries
  - > src/main/java 1214
  - JRE System Library [JavaSE-1.8]
  - build 1214
  - domains 1214
    - esslli 1214
      - bml 1214
      - generation 1214
      - json 1214
        - answer.json 1214
        - feedback.json 1214
        - intent.json 1214
        - mood.json 1214
        - question.json 1214
        - request.json 1214
      - opendial 1214
      - turntaking 1214
      - esslli.db 1214
      - esslli.txt 1214
      - esslliModel.txt 1214
    - README 1214
  - lib 1214
  - > src 1214
  - build.gradle 1214
  - questions.csv 1214
- inprotk 1213

nlg.xml | behaviour.xml | intent.json | hello.xml | EsslliListener. | Main.java | config.xml | feedback.json

```
1 {
2     "intents": [
3     { "name":"root",
4       "children": [
5        {"child":"master"},
6        {"child":"contestant"},
7        {"child":"mood"},
8        {"child":"greet"}
9        ]
10    },
11    { "name":"greet",
12      "children": [
13      ],
14      "properties": [
15       {"property":"hello"}
16      ]
17    },
18    { "name":"master",
19      "children": [
20       {"child":"question"},
21       {"child":"feedback"}
22      ],
23      "properties": [
24       {"property":"master"}
25      ]
26    },
27    { "name":"contestant",
28      "children": [
29       {"child":"answer"},
30       {"child":"request"}
31      ],
32      "properties": [
33       {"property":"contestant"}
34      ]
35    }
```

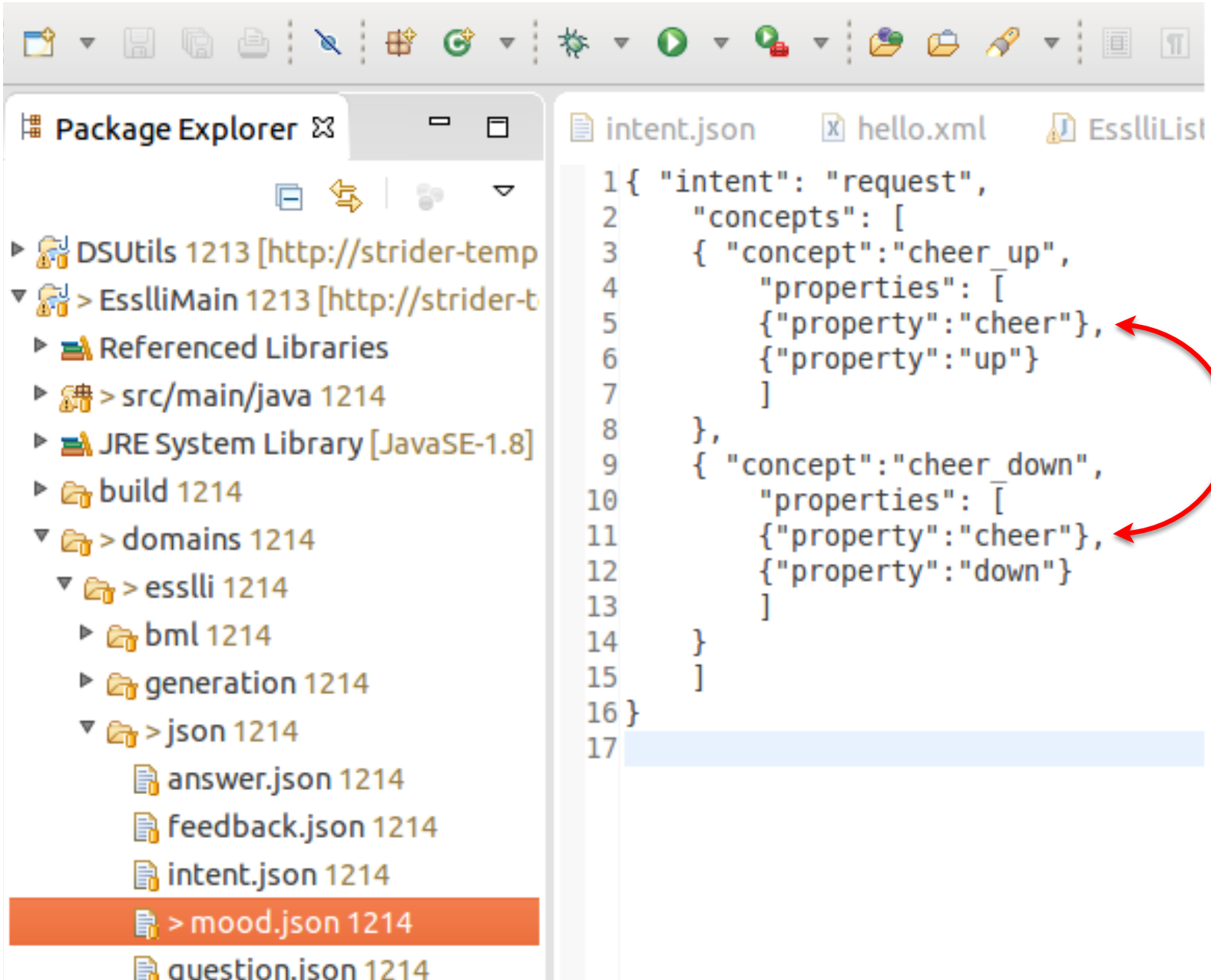
keyword „hello“ activates concept „greet“

concept „master“ branches into other files
(loaded automatically at start-up)

Problems | Javadoc | Declaration | Console | Tasks

<terminated> Main [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (12.08.2016 10:54:07)

Modules: iNLU



conflict between concepts after saying „cheer“
(resolved at runtime in context, e.g. other properties)

Modules: DM

Variable declaration

***trigger* variable changes, fire first case of every rule (think if-then-else-if-…)**

value check on confidence of concept indicated by variable „SlotIU“

value check on concept itself

setting variable value (used by another module)

special variable *feedback*: realized asap

special variable *decision*: realized as soon as floor is taken

nlg.xml | behaviour.xml | nod.xml | esslli.xml | intent.json

```
15          </variable>
16          <variable id="happiness">
17              <value>0.3</value>
18          </variable>
19          <variable id="interrupt">
20              <value>false</value>
21          </variable>
22      </initialstate>
23
24      <model trigger="SlotIU">
25          <rule id="mood">
26              <case>
27                  <condition>
28                      <if var="SlotIU" relation="&gt;" value="0.75" />
29                      <if var="SlotIU" relation="in" value="cheer_up" />
30                  </condition>
31                  <effect>
32                      <set var="happiness" value="{happiness}+0.4" />
33                  </effect>
34              </case>
35              <case>
36                  <condition>
37                      <if var="SlotIU" relation="&gt;" value="0.75" />
38                      <if var="SlotIU" relation="in" value="cheer_down"
39                  </condition>
40                  <effect>
41                      <set var="happiness" value="{happiness}-0.4" />
42                  </effect>
43              </case>
44          </rule>
45
46          <rule id="ruleSelect">
47              <case>
48                  <condition>
49                      <if var="SlotIU" relation="&gt;" value="0.75" />
50                  </condition>
51                  <effect>
52                      <set var="feedback" value="nod" />
53                  </effect>
54              </case>
55          </rule>
56
57          <rule id="quiz">
58              <case>
59                  <condition>
60                      <if var="SlotIU" relation="&gt;" value="0.75" />
61                      <if var="SlotIU" relation="in" value="greet" />
62                  </condition>
63                  <effect>
64                      <set var="decision" value="greet" />
```

Modules: NLG

Package Explorer

- DSUtils 1213
- > EsslliMain
  - Referenced Libraries
  - > src/main/java 1214
  - JRE System Library [JavaSE-1.8]
  - build 1214
  - > domains 1214
    - > esslli 1214
      - bml 1214
      - generation 1214
        - behaviour.xml 1214
        - nlg.xml 1214
      - > json 1214
      - > opendial 1214
      - turntaking 1214
      - esslli.db 1214
      - esslli.txt 1214
      - esslliModel.txt 1214
    - README 1214
  - lib 1214

nlg.xml | behaviour.xml | nod.xml | esslli.xml | intent.json | hello.x

```xml
<nlg>
    <decision value="greet">
        <action bml="hello.xml"/>
    </decision>
    <feedback value="nod">
        <action bml="nod.xml"/>
    </feedback>
    <decision value="requestQuestion">
        <action utterance="ok, what's the question?"/>
    </decision>
    <decision value="repeat">
        <action method="repeat()"/>
    </decision>
    <decision value="gloom">
        <action utterance="damn"/>
    </decision>
    <decision value="cheer">
        <action utterance="hurray"/>
    </decision>
    <decision value="offerQuestion">
        <action method="startQuestion()"/>
    </decision>
    <decision value="offerNextTip">
        <action method="continueQuestion()"/>
    </decision>
    <decision value="tryAnswer">
        <action method="offerAnswer(salientAnswer)"/>
    </decision>
    <decision value="checkAnswer">
```

BML behaviors

Package Explorer

- DSUtils 1213
- > EsslliMain 1213
  - Referenced Libraries
  - > src/main/java 1214
  - JRE System Library [JavaSE-1.8]
  - build 1214
  - > domains 1214
    - > esslli 1214
      - > bml 1214
        - > hello.xml 1214

nlg.xml | behaviour.xml | nod.xml | esslli.xml | intent.json | hello.xml | EsslliListener. | Main.java

```xml
<bml xmlns="http://www.bml-initiative.org/bml/bml-1.0" id="bml1" xmlns:bmlt="http://hmi.ewi.utwente.nl/bml">
    <speech id="s1">
        <description priority="1" type="application/ssml+xml">
            <speak xmlns="http://www.w3.org/2001/10/synthesis">Hello World.</speak>
        </description>
    </speech>
</bml>
```

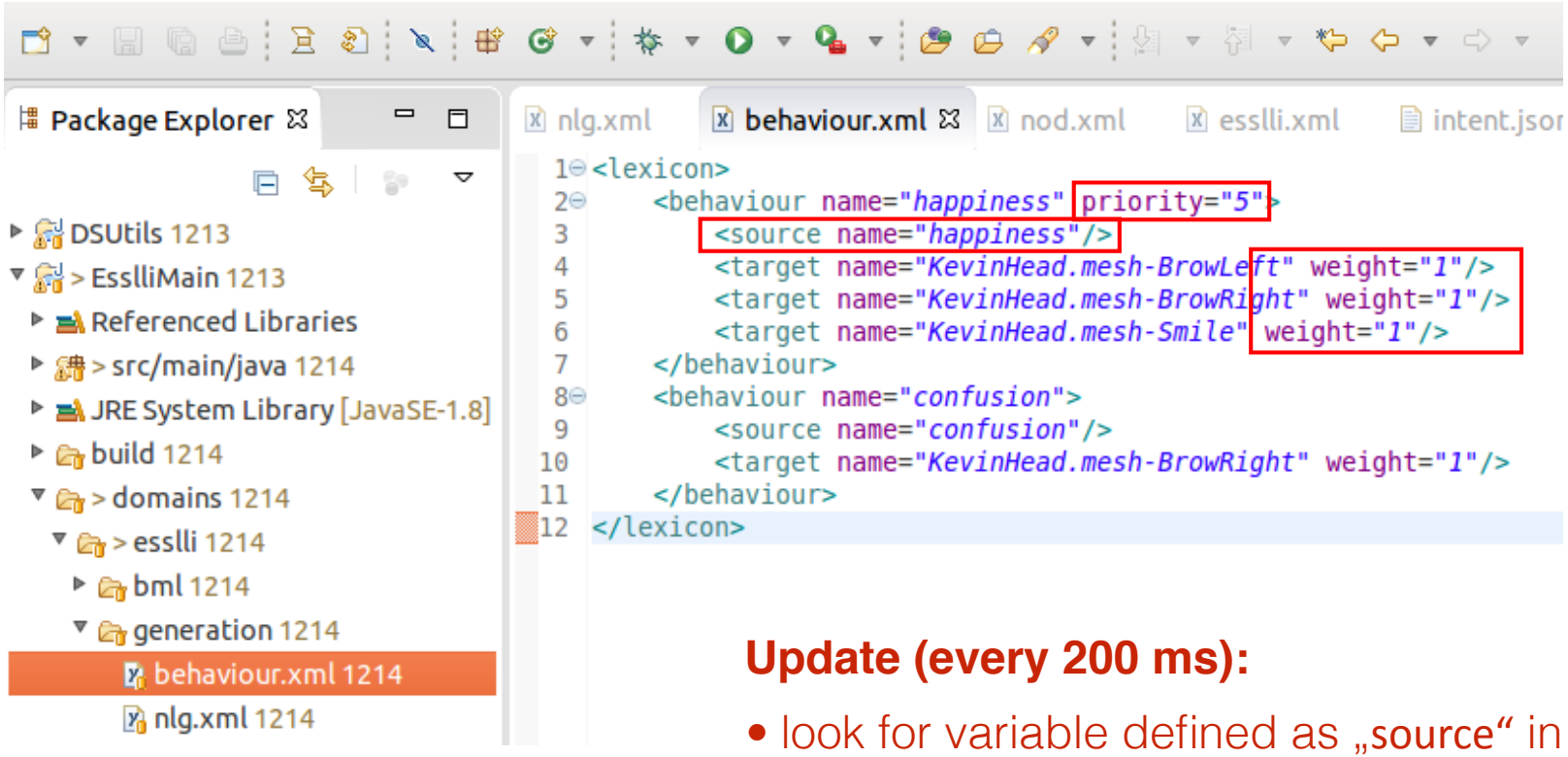
Billie says: „Hello World“

nlg.xml | behaviour.xml | nod.xml | esslli.xml | intent.json | hello.xml

```xml
<bml xmlns="http://www.bml-initiative.org/bml/bml-1.0"  id="bml1"  >
    <head id="nod1" lexeme="NOD" amount="0.1"/>
</bml>
```

Billie nods

Modules: NVBG



Update (every 200 ms):

- look for variable defined as „**source**“ in *dialogstate*
- use value and defined „**weight**“ to calculate next blendshape
- send BML block to ASAP

Conflict resolution

- lower priority behavior gets dropped
- priorities can change at runtime

Example:
facial expressions

Modules: Turn-taking

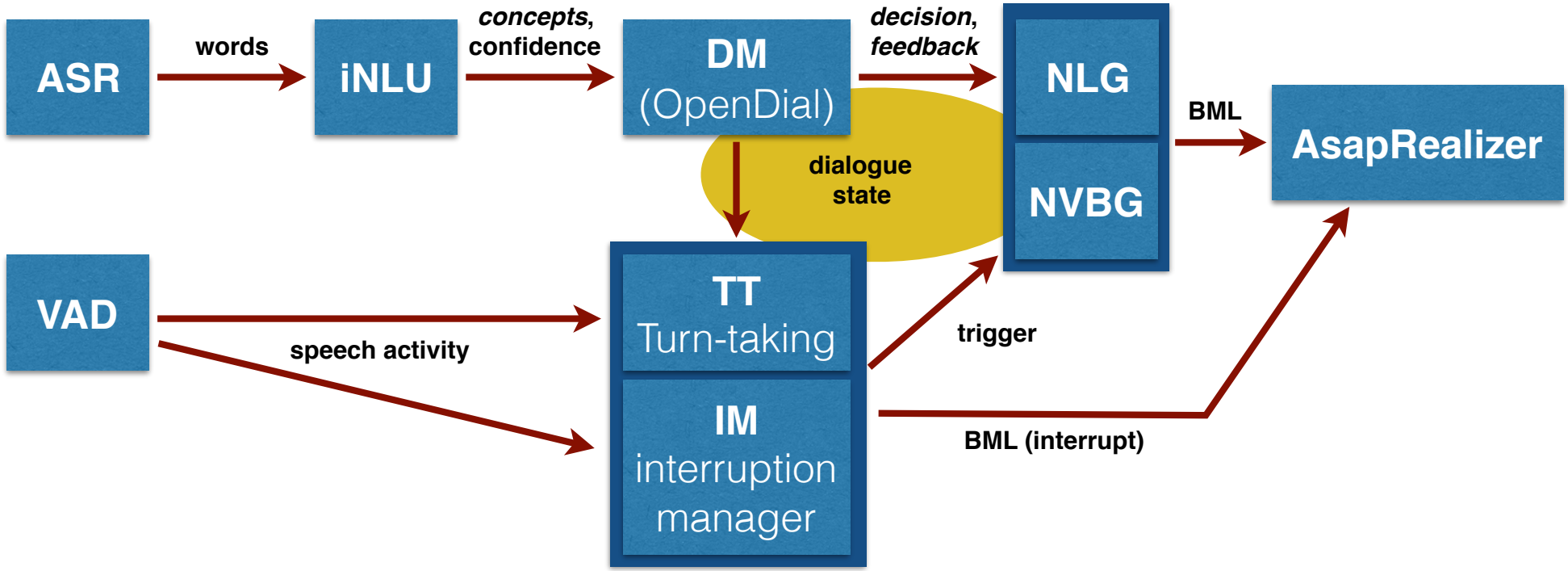
nlg.xml behaviour.xml nod.xml esslli.xml int

```
1 <turntaking>
2     <param name="delay" value="500" />
3     <param name="silenceTolerance" value="10000" />
4 </turntaking>
```

„delay“ = time in ms the system waits after a silence detection (VAD), before carrying out a pending *decision*

„silenceTolerance“ = time before „default“-action (defined within NLG) is carried out in case of total silence

Modules (summary)



Questions?

Input: „Hello“ + „I want to be contestant.“

Problems @ Javadoc Declaration Console Tasks

Main [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (12.08.2016 11:25:19)

```
0       asrword : {word=hello}
20      inlu    : {confidence=0.9999650012599576, concept=greet, intent=intent}
100     feedb.  : {feedback=nod}
103     dm      : {decision=greet, confidence=0.9999650012599576, entity=greet}
106     nlg     : {bml=greet, executedAction=greet}
161     tts     : {file=/tmp/bml1-s13748599255987377272.wav, phonems=[(0)(0)(0)]#[(h)(0.011)(0.131)][(@)(0.131)(0.173)][(l)
15876   asrword : {word=i}
16838   asrword : {word=want}
17269   asrword : {word=to}
17811   asrword : {word=be}
17816   inlu    : {confidence=0.315332625845156, concept=question8, intent=question}
19961   asrword : {word=contestant}
19964   inlu    : {confidence=0.9999650012599576, concept=contestant, intent=intent}
19976   feedb.  : {feedback=nod}
19987   dm      : {decision=offerQuestion, confidence=0.9999650012599576, entity=contestant}
19992   nlg     : {executedAction=offerQuestion, utterance=this composer was given the task of improvising on a royal theme
20265   tts     : {file=/tmp/bml3-s13211232691220016298.wav, phonems=[(0)(0)(0)]#[(D)(0.011)(0.061)][(I)(0.061)(0.111)][(s)
```

?

ms since 1st word uttered or VAD calibration finished

Q: what happens at 17816 ms?

Source code

mood.json | question.json | request.json | turntaking.xml | GoogleASR.java | TreeModule.java | *InterruptManag

```
390         @Override
391         public void run() {
392
393             try {
394                 BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
395                 String decodedString;
396                 while (!inShutdown && (decodedString = in.readLine()) != null) {
397
398                     // FLAG: ASR(raw)
399                     LocalMessageIU localIU = new LocalMessageIU();
400                     localIU.setCategory("asrraw");
401                     localIU.getPayload().put("asrresult", decodedString);
402                     if(!decodedString.equals("{\"result\":[]}"))
403                         outBuffer.add(localIU);
404
405                     processJSON(decodedString);
406                 }
407                 terminateDump();
408             } catch (Exception e) {
409                 // con.disconnect();
410                 // throw new RuntimeException(e);
411             } finally {
412                 con.disconnect();
413             }
414         }
415     }
```

Important places marked (flagged), e.g. „ASR(raw)“

Problems | Javadoc | Declaration | Console | Tasks

11 items

✓	!	Description	Resource	Path	Location	Type
	!	FLAG: ASR rightBuffer	GoogleASR.java	/inprotk/src/inpro/i	line 584	Java Task
	!	FLAG: ASR(raw)	GoogleASR.java	/inprotk/src/inpro/i	line 398	Java Task
	!	FLAG: event reactions	EsslliListener.java	/EsslliMain/src/mai	line 41	Java Task
	!	FLAG: Interrupt Behaviour (Agent -> User)	InterruptManager.java	/EsslliMain/src/mai	line 68	Java Task
	!	FLAG: Interrupt Decision	InterruptManager.java	/EsslliMain/src/mai	line 26	Java Task
	!	FLAG: Interrupt Handling (User -> Agent), gets called from VAD	InterruptManager.java	/EsslliMain/src/mai	line 34	Java Task
	!	FLAG: NLU leftBuffer	INLUModule.java	/EsslliMain/src/mai	line 71	Java Task
	!	FLAG: NLU rightBuffer	INLUModule.java	/EsslliMain/src/mai	line 148	Java Task
	!	FLAG: TurnTaking decision	TurnTakingManager.java	/EsslliMain/src/mai	line 60	Java Task
	!	FLAG: TurnTaking notifier	NLGModule.java	/EsslliMain/src/mai	line 105	Java Task
	!	FLAG: VoiceActivityDetection	GoogleASR.java	/inprotk/src/inpro/i	line 294	Java Task

Task 1

1. Type „hello“ in the text input window (Exc. 4a)

   - See what happens

   - Understand what happens — look at: json/intent.json, opendial/esslli.xml, generation/nlg.xml, bml/hello_wave.xml

   - Extend the ways Billie can be greeted!

   - Extend the way Billie reacts!

Task 2

2. Add responsive feedback behavior (Exc. 4b)

   - Open the file opendial/esslli.xml and find the rule "feedback". This rule makes Billie nod every time the system understood a concept with a confidence of over 75%.

   - The variable “feedback” triggers immediate execution of an action.

   - The feedback action is defined in behaviour/nlg.xml and bml/nod.xml

   - Make him say “ok” when his confidence is higher than 85% !

   - Test with the text window by typing in "hello?" and Enter. Note, that appending a question mark to a word indicates that the turn is kept (i.e. there is more to come).

Task 3

3. Make Billie happier (Exc. 4c)

- Open the files json/intent.json and mood.json

- Find the child called “mood” of the intent "root". This child enables the system to parse the mood.json file, in which there’s a concept called “cheer_up“ and “cheer_down”.

- Open the file opendial/esslli.xml and find the rule "mood".

- See how giving one of those concepts either increases or decreases the variable "happiness". In this case, the condition to have a confidence over 0.75 is important, because when you type or say "cheer", the system will give both concepts a probability of about 50% as the property appears in both concepts.

- Make Billie get happier when he gets praised!