IWAKI INTERACTION MANAGER
A MANUAL

MAXIM MAKATCHEV

May 30, 2013

Copyright © 2013 by Maxim Makatchev. All rights reserved.

Maxim Makatchev: *IWAKI Interaction Manager . A Manual*
©2013

1 INTRODUCTION

Iwaki Interaction Manager is an open-source dialogue manager that is not limited to dialogue. It is most closely related to the *SharedPlan* theory of discourse [Grosz and Sidner, 1990; Lockbaum, 1998], and its implementation, COLLAGEN [Rich and Sidner, 1998]. From the interaction pespective, it is designed to enable:

- Multi-modal interaction. Iwaki actions can be be interpreted by action generators of any nature.
- Multi-party interaction. Iwaki has the functionality to keep the context related to each individual interactor.
- Flexible turn taking and mixed initative. As a dialog manager, Iwaki can take initative in turn taking based on the system state, such as time. Similarly, Iwaki can process a user utterance produced at any time.
- Flexible discourse structure. Iwaki can respond to the user's utterance that is out of the current context.

From the interaction designer's perspective, Iwaki differs from some other dialogue managers in that:

- Interaction designer and content developer do not need to write code. Iwaki scripts are XML that can be generated via web forms or by hand.
- Iwaki does not make you choose between information-state and a script-based dialogue management. To an extent, you can have both.
- Iwaki is designed to play well with other developing standards for interaction design, such as BML.
- Iwaki is soft real-time. Iwaki scripts can be time-sensitive, hense it likes to be called periodically and on time.

From the software architecture prospective, Iwaki is a C++ library that can be called periodically from your interaction module to update its state and process inputs and outputs. In a complex application, however, to ensure timely periodic calls to the interaction manager, Iwaki library would likely be wrapped into a separate executable and interface with the rest of the system via an interprocess communication.

1.1 RELATED WORK

Collaborative agent-based approach to dialogue management is only one of several approaches to dialogue management (see an overview by Bui [2006]). Previously, the definitive implementation of the *SharedPlan* model [Grosz and Sidner, 1990] of collaborative discourse was the Collagen collaboration manager [Rich and Sidner, 1998], which was proprietary. An open source successor of Collagen, called Disco [Rich and Sidner, 2012], is currently under active development. An extension of SharedPlans that explicitly represents the task structure, called Collaborative Problem-Solving [Blaylock and Allen, 2005], has been used in another proprietary dialogue manager, SAMMIE Becker et al. [2006]. In the development of the IM, we follow the main ideas presented in Collagen and Disco and adapt them for our specific needs.

1.2 OVERVIEW

Similar to Collagen and Disco, IM creates and maintains a dialogue tree that represents the current state of the dialogue. At present, there is no separate task structure. According to the collaboration discourse theory, a dialogue is viewed as a hierarchy of tasks, and an utterance can contribute to a task in one of the three ways: (1) provide a needed input, (2) select a new task or subtask to work on, or (3) select a recipe to achieve the task. Like Collagen and Disco, IM extends this interpretation paradigm to include inputs of other modalities.

The tree-growing process is a production-like system (c.f. CLIPS expert system [Giarratano and Riley, 1998]) that attempts to satisfy goals by backchaining on rules. All the inputs and state changes are interpreted in the context of the current dialogue tree. According to the joint plan theory of interaction, every participant of the interaction contributes to building the joint plan. Generally, each participant may have their own view of the joint plan, distinct from others. IM considers only one view of the joint plan, which can be considered the robot's point of view.

The knowledge that comes from outside of IM (facts) is represented as logical formulae. In the simple case, these facts form a conjunction of *atoms*, where each atom is a partially grounded predicate.

IM *recipes* have a function similar to Disco and Collagen recipes, and correspond to the rules of a production system. A recipe consists of a precondition, a body and a postcondition. A precondition is a disjunctive normal form (DNF) of partially grounded atoms. A body of the recipe is an ordered sequence of *actions*, *assignments*, or *goals*. The postcondition of a recipe is an assignment.

The are several ways in which IM differs from a typical production system:

- Items in the body of a recipe (corresponding to the right hand side of a production rule) are executed in order (sequentially by default). The execution proceeds to the next item when execution of the previous item is successfully completed.
- There is a mechanism to control the flow of execution depending on the return status of actions. For example, if a time-sensitive action, such as saying "goodbye" to a departing user, has been aborted by the executor, for some reason, the IM may need to purge the rest of the farewell recipe. If that was a greeting, the IM may retry sending it to the behavior executor, or skip it and move to the next action in the greeting recipe.
- There is a mechanism for timing out of items in the body of a recipe.
- There is a whilecondition, failure of which triggers purging of the recipe.

IM's dialogue tree interface implements the three ways of interpretation of an input specified by collaborative discourse theory. Notably, IM allows interleaved recipe execution, resulting in simultaneous multi-topic dialogues. However, at the present state, IM does not implement many of the features of Collagen and Disco, such as a *focus stack*.

1.3 APPLICATIONS

Iwaki interaction manager has been originally developed to control interaction of social robots at the Robotics Institute of Carnegie Mellon University. It powers such robots as Gamebot, the scrabble-playing trash-talking robot, and Hala 2, the bilingual robot receptionist at CMU Qatar [Simmons et al., 2011]. It has also been used as a rather advanced soundboard to generate dialogue for a radio show sidekick, called Uncle Georgi. The soundboard application code is provided with Iwaki distribution as an example.

1.4 LICENSE

Iwaki Interaction Manager is distributed under the conditions of GPLv3. Copyright © 2009–2013, Maxim Makatchev, Reid Simmons, Carnegie Mellon University.

For a copy of the GNU General Public License see `http://www.gnu.org/licenses/`.

2 QUICK START

This chapter intends to briefly get the reader up to speed with an example IM installation. There are three types of files that IM needs to find in its scripts directory: the configuration file, recipe files in `SCRIPTROOT/recipes` subdirectory, and action definition files, in `SCRIPTROOT/actions`. The location of the scripts directory, as well as the name of the configuration file, can be specified as command line arguments to the IM executable, `imcore`.

2.1 RUNNING A STANDALONE PROCESS

Command: `imcore`

Command line options:

`-h` Prints command line options information

`-d level` One of the following debug levels: `ERROR`, `WARNING`, `INFO`, `DEBUG`, `DEBUG1`, `DEBUG2`, `DEBUG3`, `DEBUG4`

`-t timer_period` If present, this option causes IM to run in timer mode, where ever timer_period seconds IM state update is forced.

`-l path/log_file_prefix` If present, this option causes all the debug output be directed to the file prefixed

`-x` If present, this option enables the text-based status of IM displayed in the current terminal window.

`-p path` If present, override the default path of IM scripts directory.

`-i init_file` If present, overrides the default init file name located in the root of IM scripts directory.

2.2 CONFIGURATION FILE

IM configuration file is located in the root of scripts directory, and by default is named `initialize_im.xml`. The default init file name can be changed by `-i init_file` command line option.

The initializer file has three sections: list of triggerable recipes, list of recipe files and list of action files—see the example below.

```
<?xml version="1.0" encoding="US-ASCII"?>

<iminit>
  <!-- these recepies are always ready to be triggered by im
      itself-->
  <triggerables>
     <recipe name="engage user name qa" max_instances="any"/>
     <recipe name="heartbeat" max_instances="1"/>
  </triggerables>

  <recipefiles>
   <file>engageuser.xml</file>
   <file>engageuser_seq.xml</file>
   <file>engageuser_name_qa.xml</file>
   <file>answer_weather_followup.xml</file>
   <file>answer_weather_followup_yes.xml</file>
   <file>answer_weather_followup_no.xml</file>
   <file>heartbeat.xml</file>
   <file>chatwithuser.xml</file>
  </recipefiles>

  <actionfiles>
    <file>defaults.xml</file>
    <file>say.xml</file>
    <file>misc.xml</file>
    <file>angry.xml</file>
    <file>common.xml</file>
  </actionfiles>
</iminit>
```

2.3 RECIPES

IM is a production system, with rules specified as recipes. A recipe consists of the following elements:

- XML header:

```
<?xml version="1.0" encoding="US-ASCII"?>
```

- A unique recipe name:

```
<recipe name="engage user name qa">
```

- A precondition, which is either a Formula (a disjunction of a few conjunctions), or a set of Atoms corresponding to a single conjunction:

```
<!-- preconditions: unification amd bindings -->
<precondition>
    <atom quantifier="exist">
```

```
      <!-- object type and subtype -->
      <slot name="type" val="im"/>
      <slot name="subtype" val="user"/>
      <!-- arguments -->
      <slot name="present" val="true"/>
      <slot name="engaged" val="false"/>
      <slot name="has_been_engaged" val="false"/>
      <!-- bindings -->
      <slot name="invite_string" rel="bind" var="
          hala_invite_string"/>
      <slot name="id" rel="bind" var="present_user_id"/>
      <slot name="this" rel="bind" var="present_user_atom"/>
    </atom>
</precondition>
```

- A whilecondition, which is either a Formula (a disjunction of a few conjunctions), or a set of Atoms corresponding to a single conjunction:

```
<!-- purge the recipe when this condition fails -->
<whilecondition>
  <atom>
    <slot name="present" val="true"/>
    <slot name="this" var="present_user_atom"/>
  </atom>
</whilecondition>
```

- A body of the recipe, which is a sequence of one of the following elements: an assignment, an action, a goal. By default the steps are performed sequentially and conditionally on the successful execution of the preceding steps.

```
<body order="sequence">
  <assignment>
          <atom>
              <slot name="engaged" val="true"/>
                <slot name="this" var="present_user_atom"/>
          </atom>
  </assignment>
  <action name="say_hello_ask_name" actor="robot"
      if_node_purged="abort"/>
  <action name="say_name" actor="user"/>
  <action name="say_nice_to_meet_you_name" actor="robot"/>
  <goal recipe_name="any" initiator="any">
    <atom>
      <slot name="uu_unhandled" val="false"/>
      <slot name="this" var="present_user_atom"/>
    </atom>
  </goal>
  <action name="say_goodbye_name" actor="robot"/>
 <!--  -->
</body>
```

- An assignpost, which is an assignment that is to be performed upon the completion of the recipe. This assignment are equivalent to the assignments in the end of the recipe body, except for they are also used in matching the recipe against the currently active goal.

```
<!-- set right after execution ends -->
<assignpost>
  <atom>
    <!-- set object which name is equal to the one stored in
         var -->
    <slot name="this" var="present_user_atom"/>
    <slot name="has_been_engaged" val="true"/>
    <slot name="engaged" val="false"/>
  </atom>
</assignpost>
```

2.4 ACTION DEFINITION FILES

Actions are the outputs of the Interaction Manager. They consist of a list of arguments, sequence of datablocks, and elements defined by the Behavior Markup Language [BML, 2012].

Here is an action with an argument and two datablocks:

```
<?xml version="1.0" encoding="US-ASCII"?>

<bml name="angry_response1">
    <roboml:args>
        <arg name="arg_utterance" type="string"/>
    </roboml:args>

    <roboml:datablock>
        <animation_id>frown1</animation_id>
        <intensity>1.0</intensity>
        <text>$arg_utterance</text>
        <focus></focus>
        <head></head>
    </roboml:datablock>

    <roboml:datablock>
        <animation_id>frown2</animation_id>
        <intensity>1.0</intensity>
        <text></text>
        <focus></focus>
        <head></head>
    </roboml:datablock>
</bml>
```

BIBLIOGRAPHY

Tilman Becker, Nate Blaylock, Ciprian Gerstenberger, Ivana Kruijff-korbayová, Andreas Korthauer, Manfred Pinkal, Michael Pitz, Peter Poller, and Jan Schehl. Natural and intuitive multimodal dialogue for in-car applications: The sammie system. In *In Proceedings of the ECAI Sub-Conference on Prestigious Applications of Intelligent Systems (PAIS 2006), Riva del Garda*, 2006. (Cited on page 2.)

Nate Blaylock and James Allen. A collaborative problem-solving model of dialogue. In *In Proceedings of the SIGdial Workshop on Discourse and Dialog*, pages 200–211, 2005. (Cited on page 2.)

BML. Behavior Markup Language 1.0 Standard. `www.mindmakers.org/projects/bml-1-0/wiki/Wiki`, July 2012. Accessed July 2, 2012. (Cited on page 8.)

Trung H. Bui. Multimodal dialogue management—state of the art. Technical report, Human Media Interaction Department, University of Twente, The Netherlands, January 2006. (Cited on page 2.)

Joseph C. Giarratano and Gary D. Riley. *Expert Systems: Principles and Programming*. Course Technology, 1998.

B. J. Grosz and C. L. Sidner. Plans for discourse. In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in communication*, pages 417–444. MIT press, 1990. (Cited on pages 1 and 2.)

K. E. Lockbaum. A collaborative planning model of intentional structure. *Computational linguistics*, 24(4):525–572, 1998. (Cited on page 1.)

Charles Rich and Candace L. Sidner. Collagen: A collaboration manager for software interface agents. *Journal of User Modeling and User-Adapted Interaction*, 8:315–350, 1998. (Cited on pages 1 and 2.)

Charles Rich and Candace L. Sidner. Using collaborative discourse theory to partially automate dialogue tree authoring. In *Proc. Int. Conf. on Intelligent Virtual Agents*, 2012. (Cited on page 2.)

R. Simmons, M. Makatchev, R. Kirby, M.K. Lee, I. Fanaswala, B. Browning, J. Forlizzi, and M. Sakr. Believable robot characters. *AI Magazine*, 32(4):39–52, 2011. (Cited on page 3.)