The example program is basically to demonstrate a simple working example of functionality of 3 layers of the architecture, namely FAtiMA, CMION and SAMGAR for a Pioneer robot.

When you run the example, Robot should approach the user and greet (Printed out from CMION competency in java “Speak Competency says Hello my name is Sarah”) the user when found in the proxemic range (this is achieved by the face detect module and needs to be adjusted (Face Percentage) inside the source code of SAMGA-FaceFinder competency for your camera).

In this example the robot should have the following:

1. OpenCV, ARIA and Yarp (version yarp-2.2.5 and above) correctly installed
2. Camera installed on the robot facing in the direction of forward motion. The camera ideally should be placed at a height where it can see faces of people standing and on the centre point of robot’s horizontal axis (between 2 wheels), since the program takes into account the position of face detected followed by the amount of turn required to face the user and approach the user.

Steps to start the example program: (This is an example program for Windows)

1. SAMGAR
   1. Run cmake (recommended cmake version 2.8) inside \lirecsvn\scenarios\TeamBuddy\SAMGAR, make sure that the paths in CMakeLists.txt points to your own packages for e.g. OpenCV and Yarp
   2. Compile the Example project it should generate some binary files. Binaries are also included in release folder.
   3. Start Yarp and SAMGAR as mentioned in SAMGAR tutorial (<http://adapsys.feis.herts.sc.uk/SAMGAR>/) for example
      1. Run yarp namespace /global followed by yarp server 20000 in command prompt window and in other window run
      2. yarp namespace /sarah followed yarp server 10000
   4. Start the SAMGAR GUI (SamgarKey)
   5. Start the modules (binaries complied from Example project)
      1. SamgarSendImage, SamgarFaceDetect: Connect them VideoOut(SamgarSendImage) to VideoIn(SamgarFaceDetect). You would see the result image in window of SAMGAR-FaceFinder module.
      2. SamgarMove, Please note that there is **no obstacle avoidance** in the motion module, motion commands directly invoke function from ARIA motion class, although the robot should move only if it finds a face. SamgarMove would connect to the robot platform
2. CMION:
3. Start java program CMION (\lirecsvn\libs\cmion\cmionLauncher), please refer the CMION doc (\lirecsvn\libs\cmion\cmion.doc) for configuring the java packages for your own programs. Also make sure that you include the package cmionTeamBuddy which contains the source code and competencies for TeamBuddy Example. Ideally you should use Eclipse to open CMION (4 projects cmionMain, cmionLauncher, cmionAddOns and ION.Meta) and then import the cmionTeamBuddy as a 5th project into the same workspace. Make sure the rest of the CMION code is up to date. In order to run the example successfully, modify the class path of cmionLauncher to include the project cmionTeamBuddy (The reason for this is that when we launch cmion we dynamically load some competencies from the TeamBuddy project, see next step)
4. Main files for CMION configuration are
   1. ArchitectureConfiguration.xml: **Make sure you copy this file** from \lirecsvn\scenarios\TeamBuddy and paste it into \lirecsvn\libs\cmion\cmionLauncher (just make a backup copy of the original file). This is the main file which sets the connection for FAtiMA and CMION. The file also specifies the XML files for CompManagerRules and CompetencyLibrary.
   2. CompetencyLibraryTeamBuddy.xml: Contains the list of competencies, is already inside the cmionLauncher folder on the svn
   3. CompManagerRulesTeamBuddy.xml: Contains the sequence and conditions for list of competencies to be started to carry out a particular mind action. Is already in the cmionLauncher folder on the svn.
5. Once you start CMION, SAMGAR GUI should show CMION competencies now and then connect the CMION-FaceDetect(In) to SAMGAR-FaceFinder(out) modules. Then connect CMION-MoveRobot(Out) to SAMGAR-Move(In) Module
6. FAtiMA
7. Go to: \fatima-bin and start TeamBuddyAgent.bat file, you would see a window open for FAtiMA mind and its status
8. In order to create this example we had to create/modify 5 main files in the \fatima-bin\data\characters\minds folder (note: the following points are just for your information, the files are already all set up correctly):
9. LIRECScenarios.xml: Where the properties about TeamBuddy scenario is specified
10. TeamBuddyActions.xml, TeamBuddyGoals.xml: Actions and goals of the agent are specified, i.e. the planning domain the agent uses
11. NoCultureTeamBuddy.xml: This file contains a culture profile, you could define rituals and social values in here. For now we don’t use this much, except for defining the praisewothiness of actions. Alternative versions of this file with modified values could be created in order to adjust our companion to a different culture
12. Inside \fatima-bin\data\characters\minds\roles\TeamBuddyAgent\ TeamBuddyAgent.xml, the personality / role of the agent is specified. Fatima was originally used as a system with multiple autonomous agents (i.e. several instances of Fatima running at once) where all agents shared the same planning domain (actions and goals), but everyone has a different role / personality. In this file therefore, we specify which goals of the goal library should be used, plus the emotional thresholds and motivation parameters that make up the agents personality. It is important to list the goals here we want the agent to carry out here, other wise the agent will not consider them.

Please start the programs in the sequence mentioned above 1) SAMGAR Modules🡪 2) CMION🡪 3) FAtiMA-TeamBuddyAgent.bat. The figure below shows the conceptual working of the example across the 3 layers.

