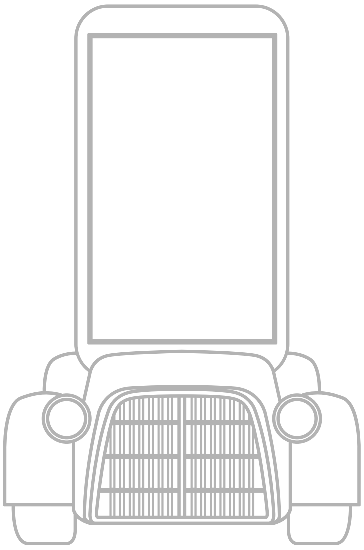
TYCHE
The AI Companion for Kids



iRSP
USER MANUAL

iRSP USER MANUAL

TYCHE

The AI Companion for Kids

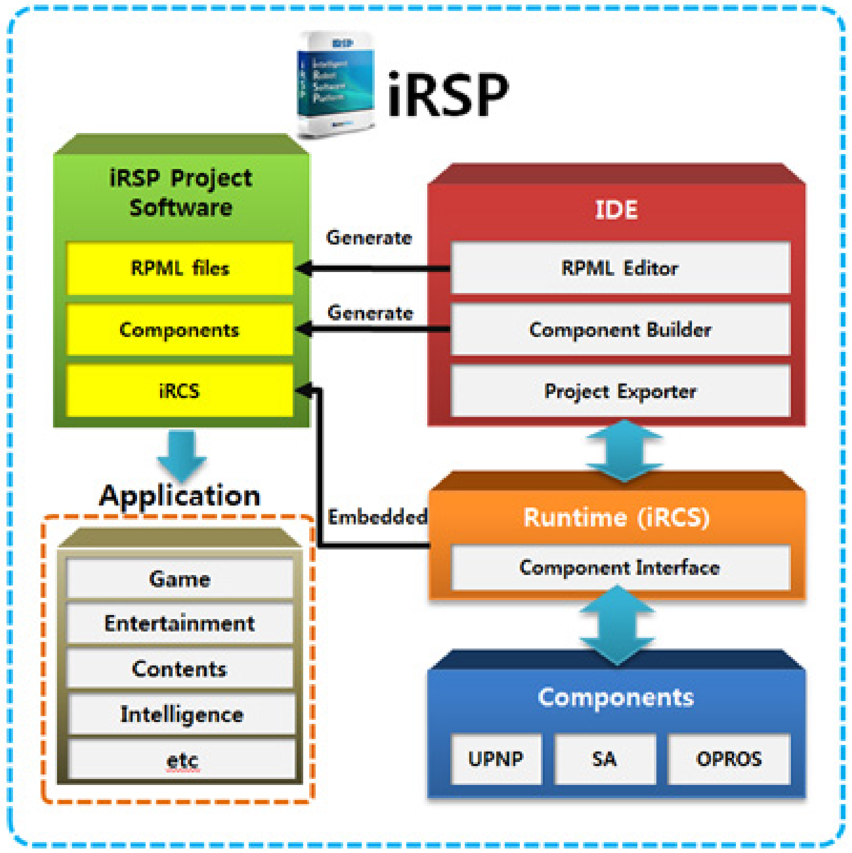
MANUAL CONTENTS

1.iRSP INTRODUCTION

1.1. What is iRSP?
iRSP is a robot software platform that helps develop and execute intelligent services required for a variety of robot systems through a robot system design step and a service integration step

1.2. iRSP architecture and features

1.2.1. iRSP System Architecture



[iRSP System Architecture]

- iRSP consists of Plan Editor, intelligent Robot Control System (iRCS) which executes a plan, and UPnP Component Builder for an UPnP Component running on a robot.

- iRSP Plan Editor provides a Visual Programming Language (VPL) environment, which enables a user to build robot programs conveniently via Drag-and-Drop. A Plan is a file that is stored in our own format called 'Robot Plan Markup Language (RPML)', and this file is used for executing a robot program.

- iRCS is a runtime module of iRSP, and it executes a plan consists of various components and nodes. It provides 'Execution', 'Stop', and 'Pause' functionalities, and all runtime errors can be verified in the plan editor.

- UPnP Component Builder is an Eclipse Plug-in based program which helps a user to develop UPnP components easily. With a component project wizard, a user can build an UPnP interface, and a template code is generated automatically. A user can program the UPnP component by filling up the template code.

1.2.2. IRSP FEATURES

Easy to Use	- Robot programming via drag-and-drop. - Intuitive plan representation with nodes and arrows.
Various Components	- Various components from low-level control components to complex components. - Provides components for external devices. - Provides algorithm components such as image processing, voice recognition.
Mobile Device Environment	- Identical UI on the mobile device environment with PC. - Robot programming with iRSP on the mobile device environment.

1.2.3. IRSP SYSTEM REQUIREMENTS

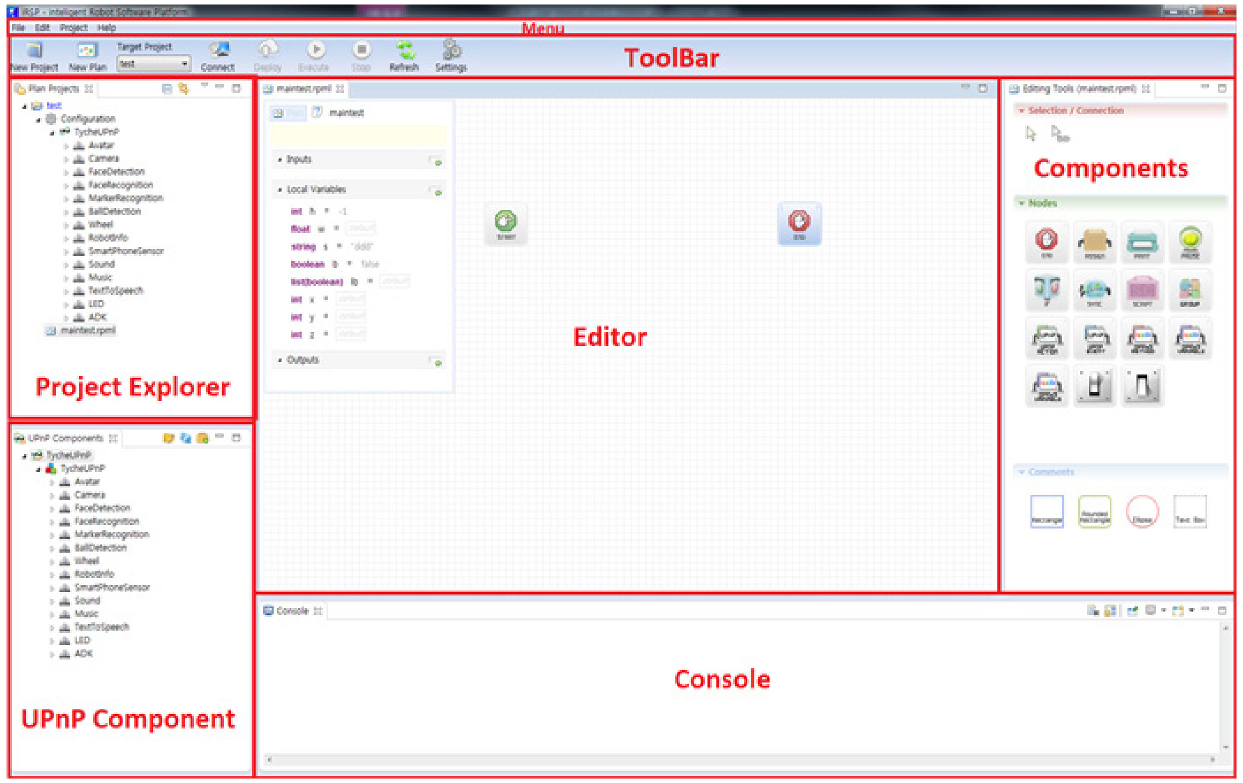
The recommended environment of iRSP is as follows.

Item	Recommended
OS	Windows XP/Vista*, Linux, Mac OS X 10.5
CPU	P4 2.0GHz
RAM	1GB
HDD	150MB (With Simulator, 1GB)
Java version	JRE 1.6

* For Windows Vista/7, an administrator rights is required.

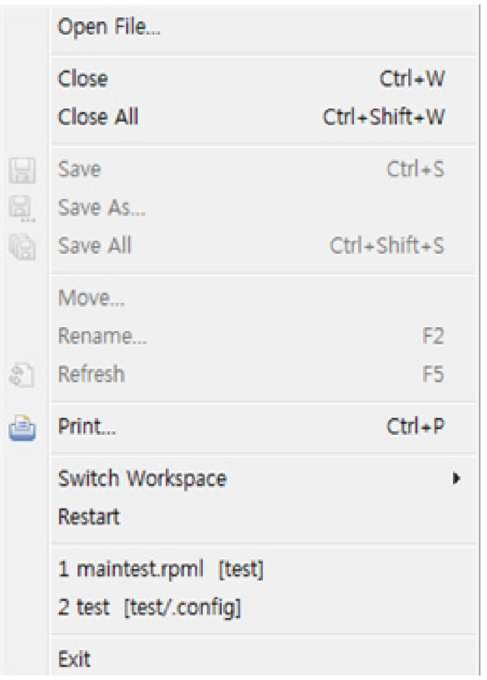
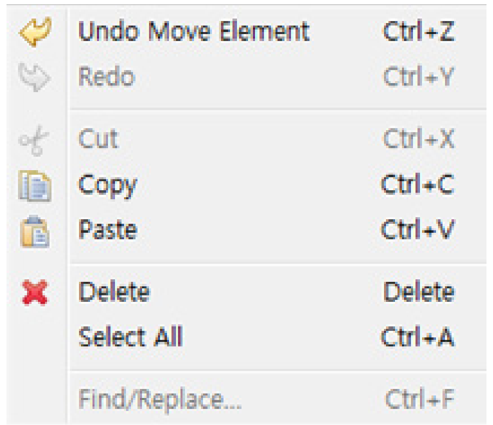
2. HOW TO USE IRSP

2.1. IRSP LAYOUT

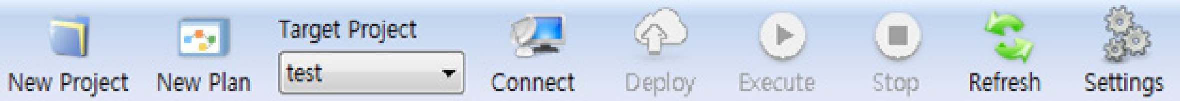


[iRSP Layout]

2.1.1. IRSP MENU

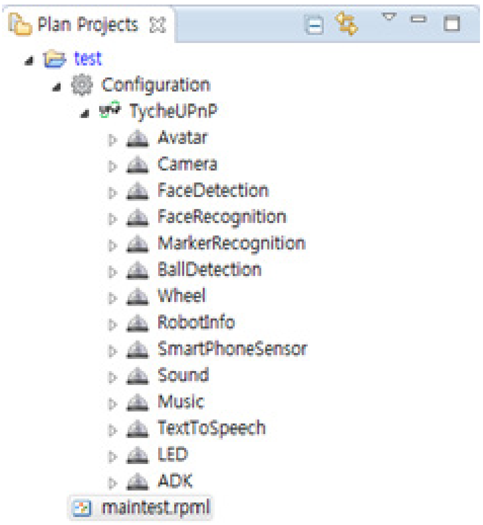
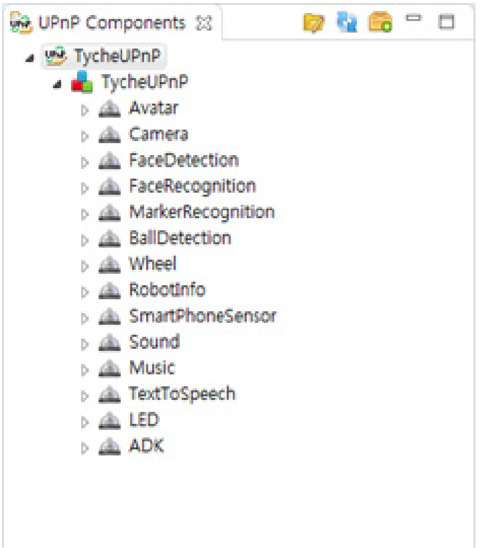
File	 Open File... Close Ctrl+W Close All Ctrl+Shift+W Save Ctrl+S Save As... Save All Ctrl+Shift+S Move... Rename... F2 Refresh F5 Print... Ctrl+P Switch Workspace Restart 1 maintest.rpml [test] 2 test [test/.config] Exit	Open File: Open the existing Plan File. Close: Close the current plan editor. Close All: Close all the plan editors. Save: Save the current plan. Save As: Save the current plan with a new file name. Save All: Save all the open plans. Move: Move the selected plan file into another project. Rename: Change the name of a project or a plan file. Refresh: Clear the results on the plan editor. Print: Print the current plan. Switch Workspace: Change the workspace path, and restart iRSP. Restart: Restart iRSP. Exit: Terminate iRSP.
Edit	 Undo Move Element Ctrl+Z Redo Ctrl+Y Cut Ctrl+X Copy Ctrl+C Paste Ctrl+V Delete Delete Select All Ctrl+A Find/Replace... Ctrl+F	Undo Move Element: Reverse the effect of recent executed job. Redo: Do the recent canceled job again. Cut: Cut the selected contents. Copy: Copy the selected contents. Paste: Paste the contents of the clipboard. Delete: Delete the selected contents. Select All: Select all contents of the current plan editor.

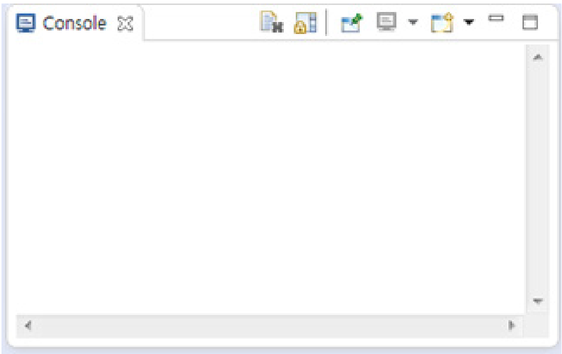
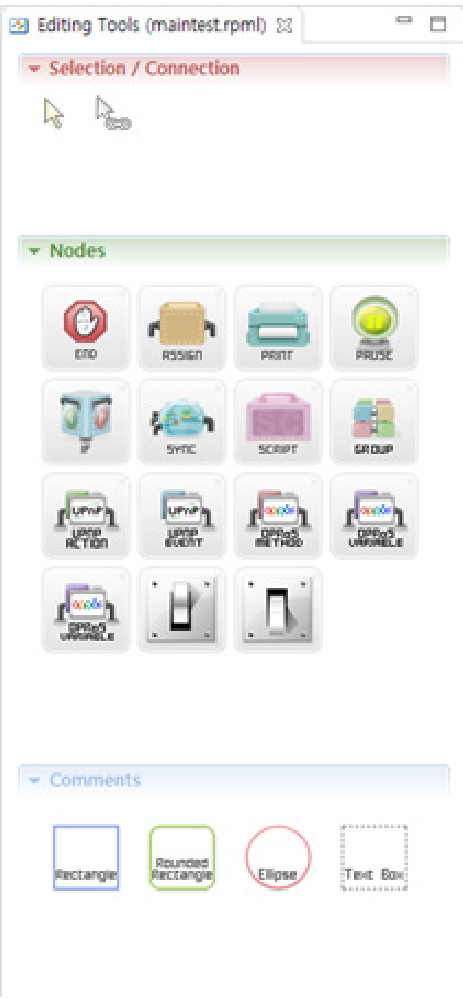
2.1.2. iRSP Toolbar



📁	New Project: Create a new project.
▦	New Plan: Create a new plan file (.rpml).
Target Project test	Execution Mode: Choose a target (Robot/Simulator) that a plan will be executed on.
🖥	Connect: Connect to the runtime
☁	Deploy: Deploy the .rpml to the runtime
▶	Execute: Run the current plan.
⏸	Pause: Stop the running plan temporarily.
■	Stop: Stop the running plan.
⟳	Refresh: Clear the results shown on the plan editor.
⚙	Setting: Set the directory for the runtime

2.1.3. IRSP VIEW

 Plan Projects test Configuration TycheUPnP Avatar Camera FaceDetection FaceRecognition MarkerRecognition BallDetection Wheel RobotInfo SmartPhoneSensor Sound Music TextToSpeech LED ADK maintest.rpml	Project: Show the all project located in the Workspace and all plan files of those projects.
 UPnP Components TycheUPnP TycheUPnP Avatar Camera FaceDetection FaceRecognition MarkerRecognition BallDetection Wheel RobotInfo SmartPhoneSensor Sound Music TextToSpeech LED ADK	UPnP Explorer: Show the list of all available UPnP Components : Refresh the list of UPnP Components : Create new folder for UPnP : Search for new UPnP

 Console	Console: Show the result of executing a plan.
 Editing Tools (maintest.rpml) Selection / Connection Nodes END ASSIGN PRINT PAUSE SYNC SCRIPT GROUP UPnP ACTION UPnP EVENT OPRoS METHOD OPRoS VARIABLE OPRoS VARIABLE Comments Rectangle Rounded Rectangle Ellipse Text Box	Components: A set of tools required for making a plan. (Selection, Nodes, Comments) A user can use it via drag-and-drop. - List of available Group nodes: Show the available Group nodes. Show the name and input/output parameters of the Group nodes. - List of available UPnP nodes: Show the available UPnP components. Show the name and input/output parameters of the UPnP components.

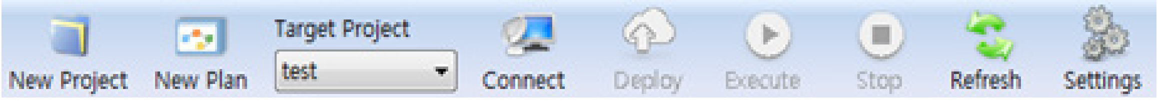
- COMPONENTS VIEW

Select		Normal Selection mode.
		This mode is for connecting nodes.
Node	END	END: Exit the plan.
	ASSIGN	ASSIGN: Store a value in a variable.
	PRINT	PRINT: Print the contents of the node in the Console View.
	PAUSE	PAUSE: Pause the plan for the amount of time specified in the input value (unit: millisecond).
	IF	IF: Branch the plan according to conditions.
	SYNC	SYNC: Only after all control flows that are connected to the node arrive, the flow can move to the next nodes.
	SCRIPT	SCRIPT: Use BeanShell script in a plan.
	GROUP	GROUP: Use the existing plan as a Group node.
Comment	Text Box	Text Box: Comment input box in the form of a borderless rectangle.
	Rectangle	Rectangle: Comment input box in the form of a bordered rectangle.
	Rounded Rectangle	Rounded Rectangle: Comment input box in the form of a rounded rectangle.
	Ellipse	Ellipse: Comment input box in the form of an ellipse.

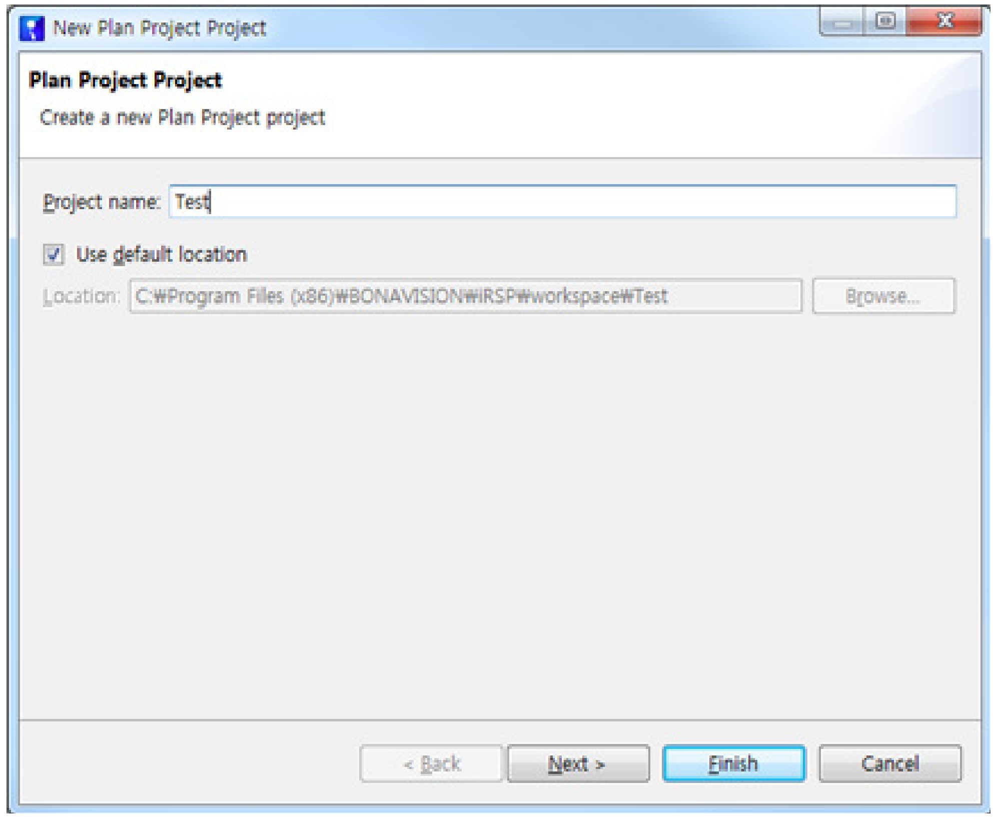
2.2.1. IRSP BASIC

2.2.2. CREATE A PROJECT

1) Click 'New Project' in the iRSP Toolbar.



2) Specify 'Project name' and Click the Next button.


New Plan Project Project

Plan Project Project

Create a new Plan Project project

Project name: Test

☑ Use default location

Location: C:₩Program Files (x86)₩BONAVISION₩iRSP₩workspace₩Test

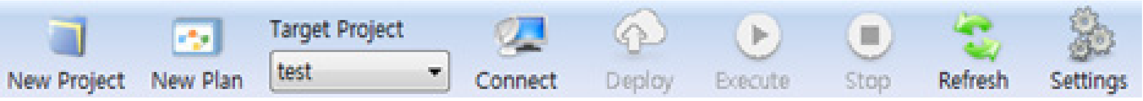
- You can change the location of workspace.

3) Within this page, you can choose a robot, and can add basic plan files, example plan files.

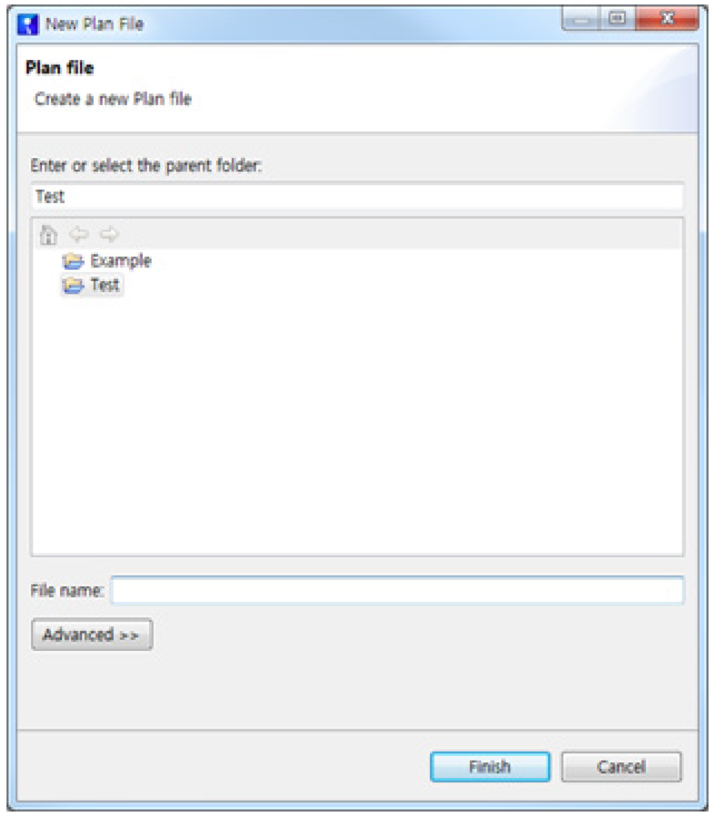
4) After clicking 'Finish' button, you can see that the required files for the project are created successfully.

2.2.3. CREATE A PLAN

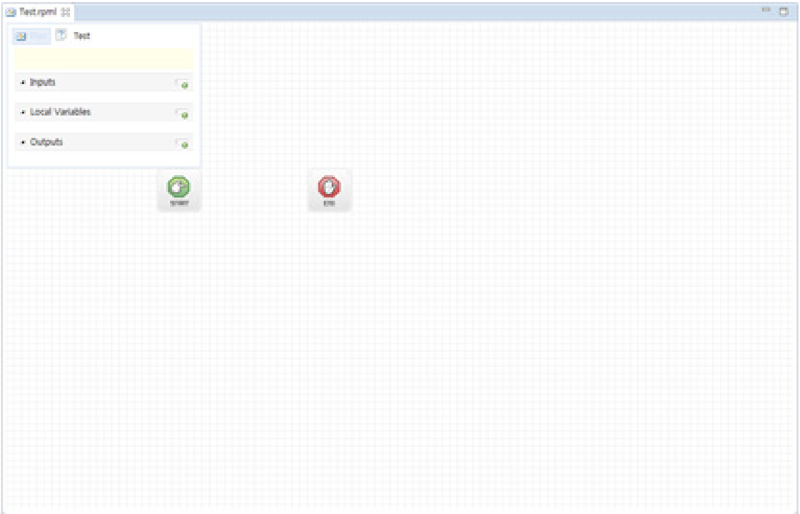
1) Click the 'New Plan' button in the iRSP Toolbar.



2) First, choose the project which contains the new plan. Then, specify the name of the plan, and click the 'Finish' button.

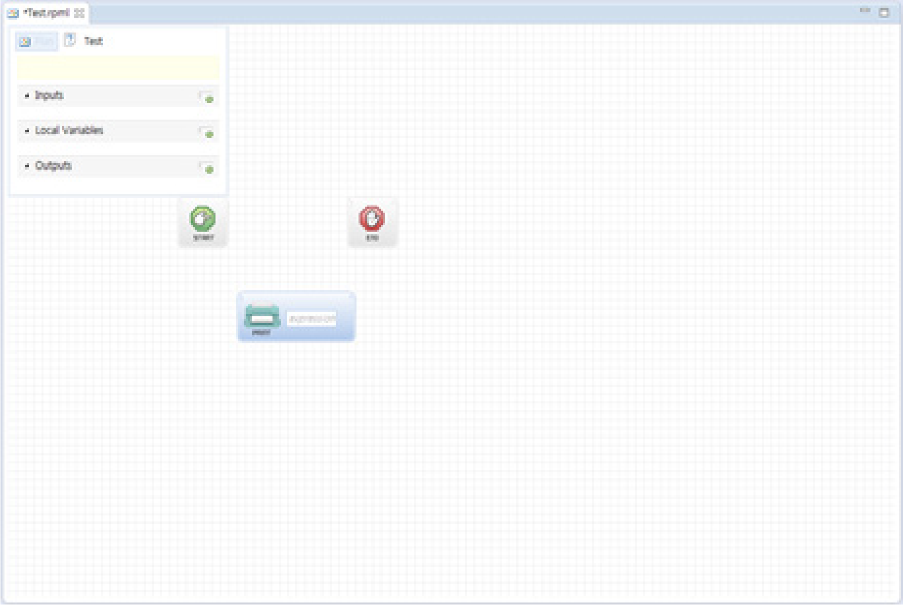


In the plan, ' Inputs', 'Local Variables', 'Outputs', 'Start node', and 'End node' are created. 'Inputs' is for the list of inputs of the plan, 'Local Variables' is the list of variables in the plan, 'Outputs' is the list of the outputs of the plan. 'Start node' is where the plan starts, and 'End node' is the end point of the plan.



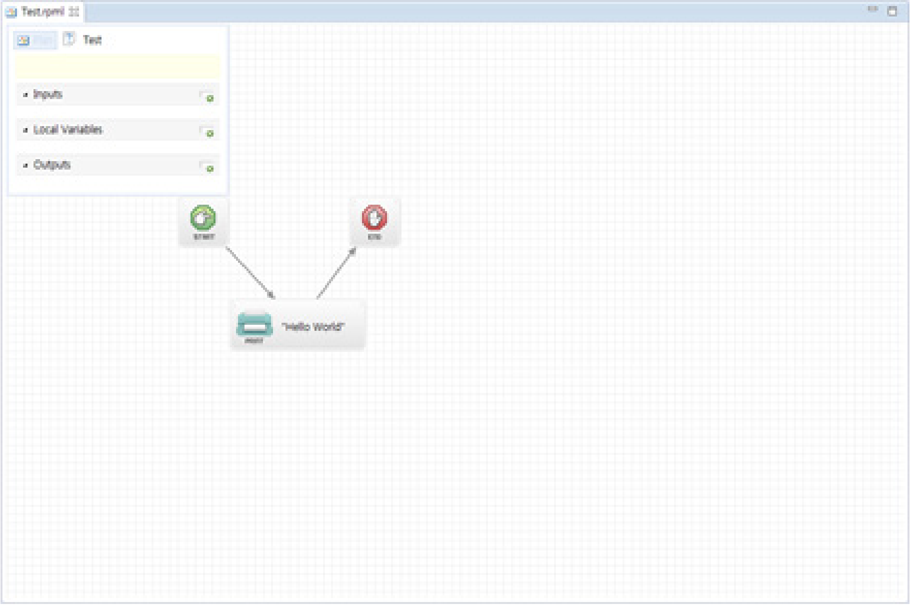
3. PRINT 'HELLO WORLD'

1) Adding a 'Print' node: Drag a 'Print' node from the 'Component View', and drop it into the 'Editor View'.



2) Entering the value into the 'Print' node: Type "Hello World" into the edit box of the 'Print' node.

3) Connecting a control flow: Connect the 'Start' node, the 'Print' node, and 'End' node in order. A control flow is represented as an arrow. You can connect nodes by dragging from the edges of each nodes or using the chain-shape 'Connection Tool ( )'.

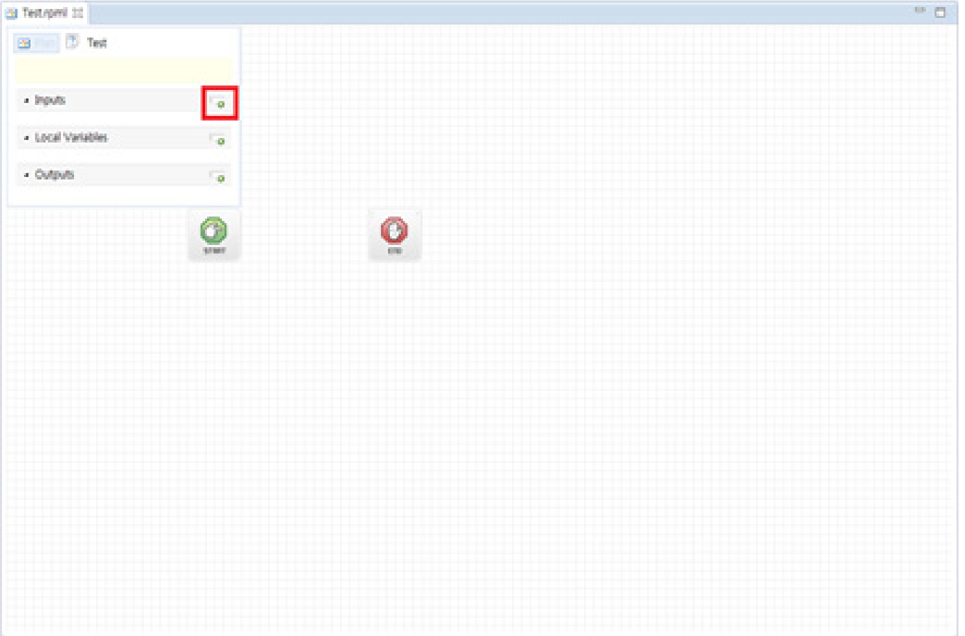


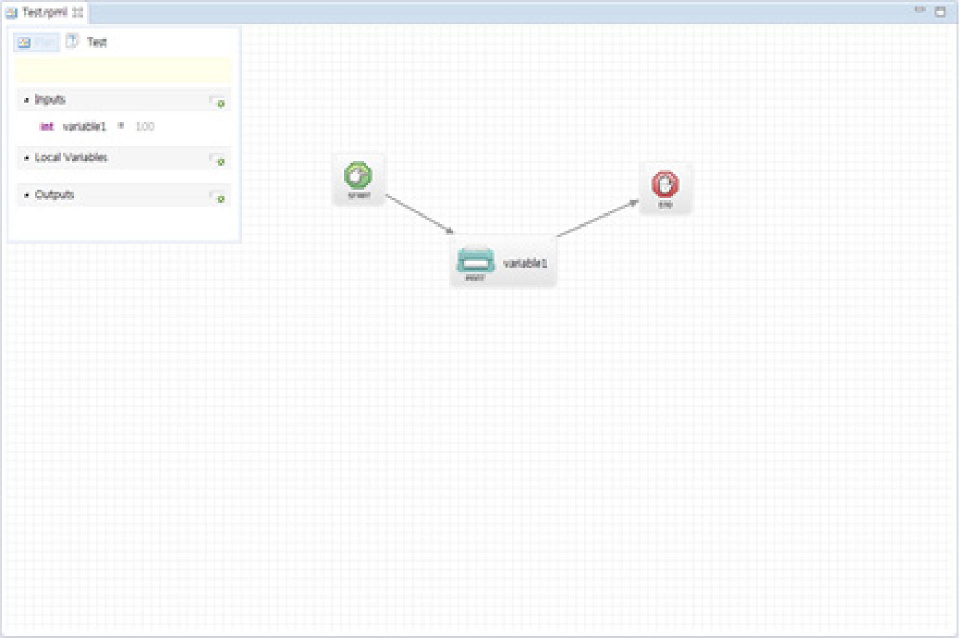
4) Execution: You can run the plan with 'Execute (▶)'. You can identify a current control flow with the highlight of nodes. You will be able to watch the "Hello World", which you typed in the edit box of the 'Print' node, in the Console View.

3.1. DECLARATION AND USE OF VARIABLES: VARIABLES

1) Adding a variable by clicking 'Add Button ( )' in the 'Variables' box located at the left-top part of the 'Editor View'.

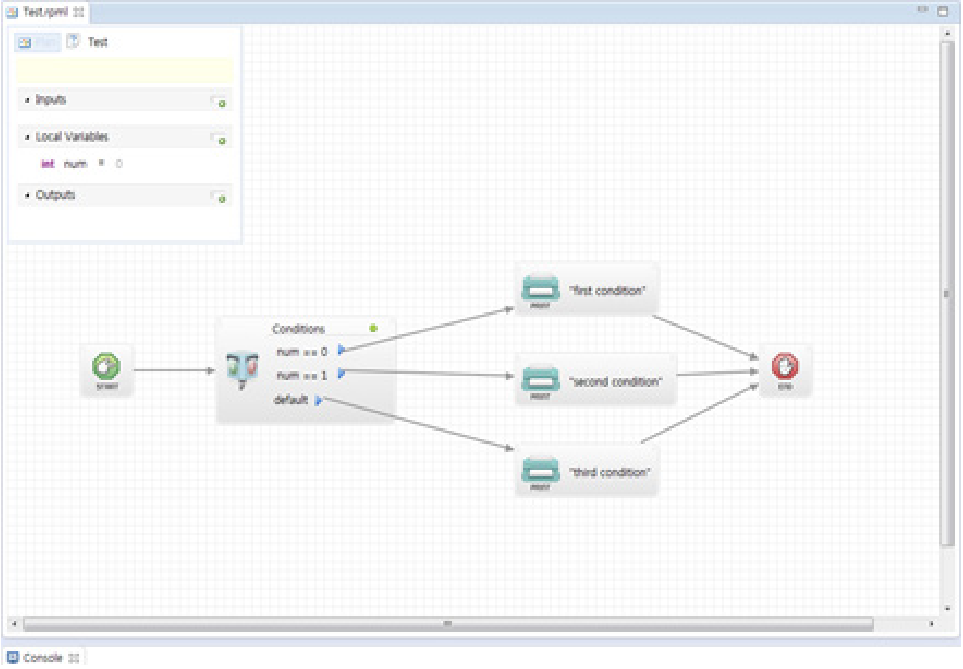
2) Specify the Type, Name, Initial value of the variable (from the leftmost box).
Declare an int-type variable 'variable1', and set the initial value of the variable as 100.





int variable1 = 100

3) To use the new variable, add a 'Print' node, and type the name of the new variable into the edit box of the 'Print' node. Connect the node with the 'Start' node and the 'End' node.
4) After you execute the plan, you can verify that the initial value of the variable is shown in the 'Console View'.

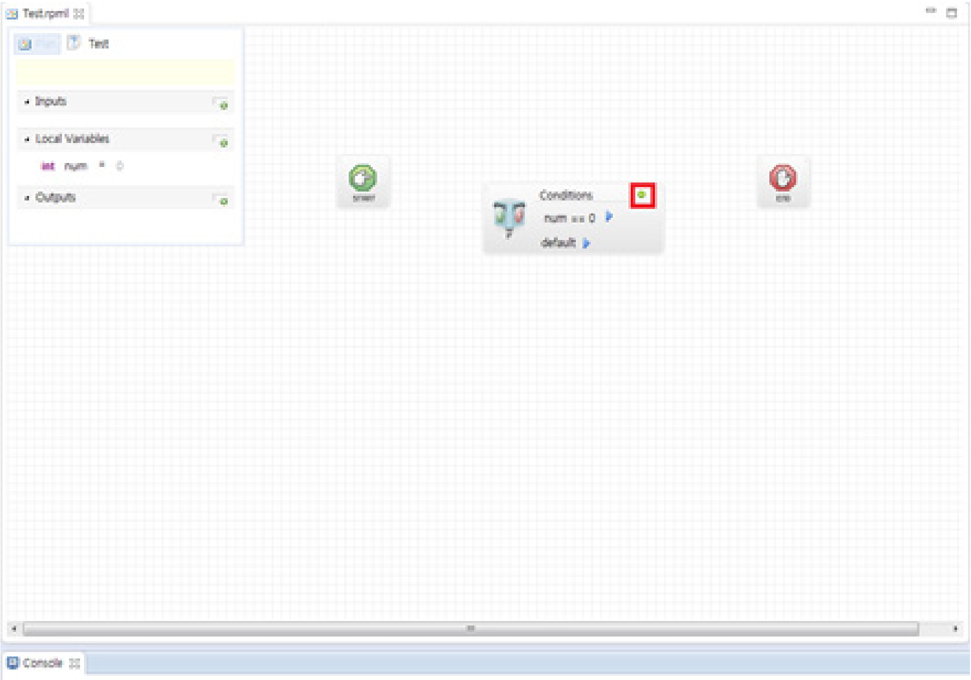


int num = 0

3.1.1. FLOW CONTROL: IF

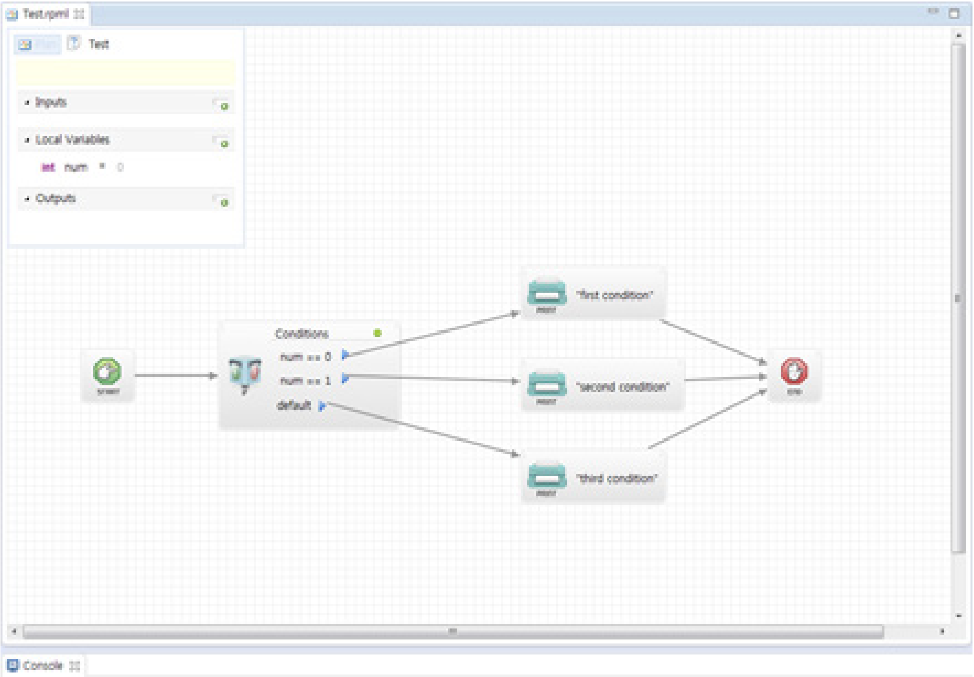
1) Adding an 'If' node: Drag an 'If' node from the 'Components View' located at the right side, and drop into the 'Editor View'.

2) Inserting a branch condition: Click the '(+)' located at the right-top of the 'If' node, and an edit box for a condition will be added. Then, specify a condition to branch in the edit box. (E.g. 'num == 0')

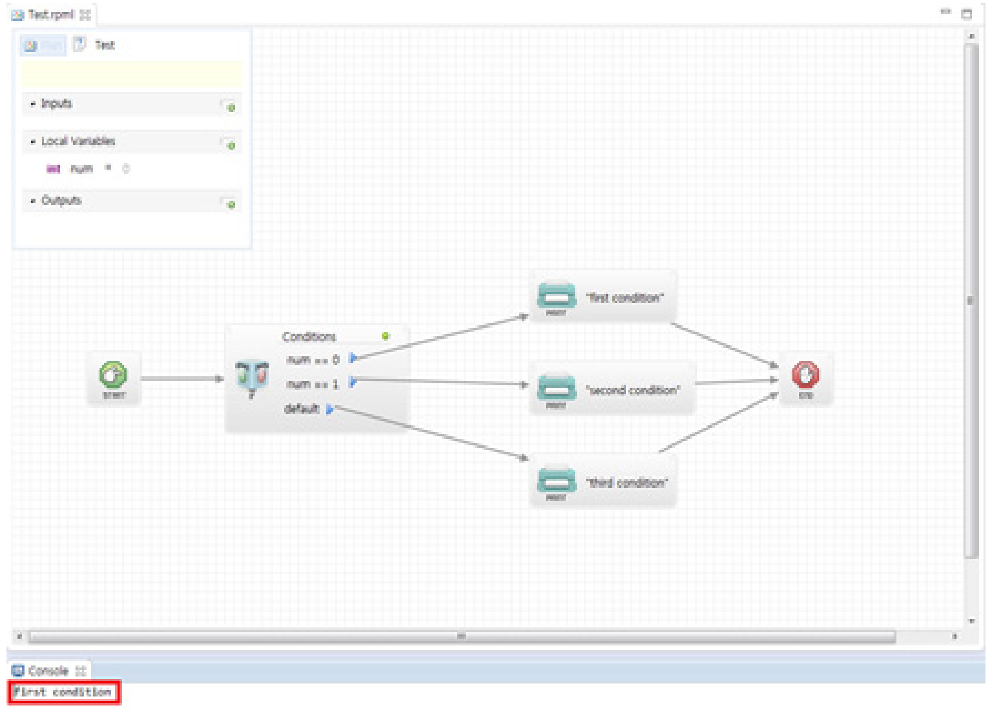


int num = 0

3) Connect the nodes that you want to execute when the condition is satisfied like the below figure. You can connect a next node from the 'Triangle (▷)' located at the right side of the condition edit box.



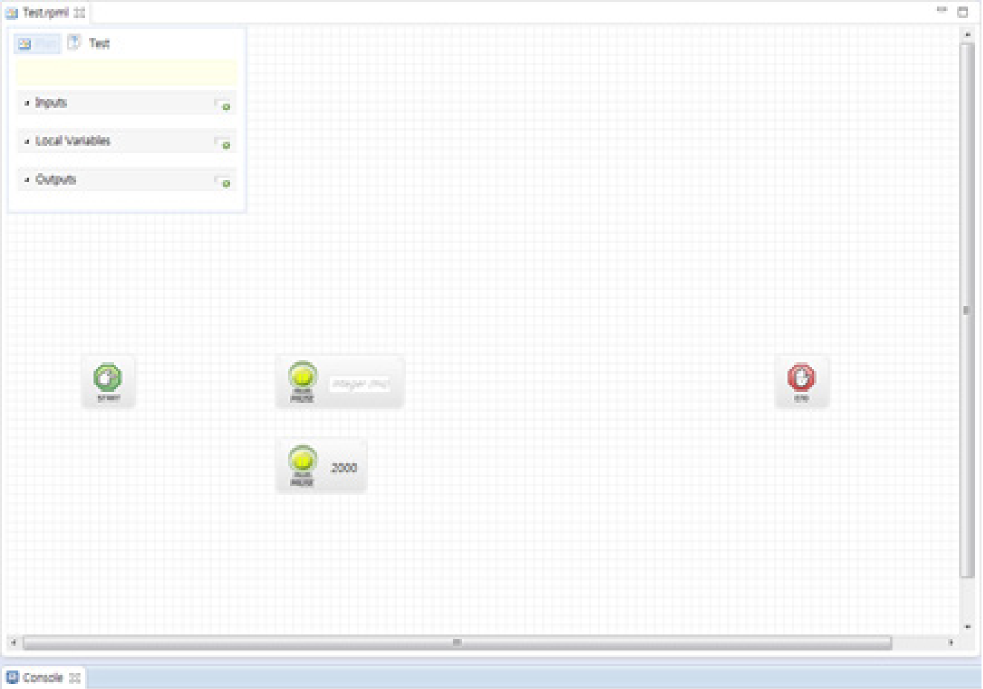
4) Execution: You can run the plan with 'Execute (▶)'. You can identify a current control flow with the highlight of nodes. The initial value of the variable 'num' is 0, and there is no change in the value of the variable. Therefore, you can watch 'first condition' in the 'Console View' according to the condition of the 'If' node.



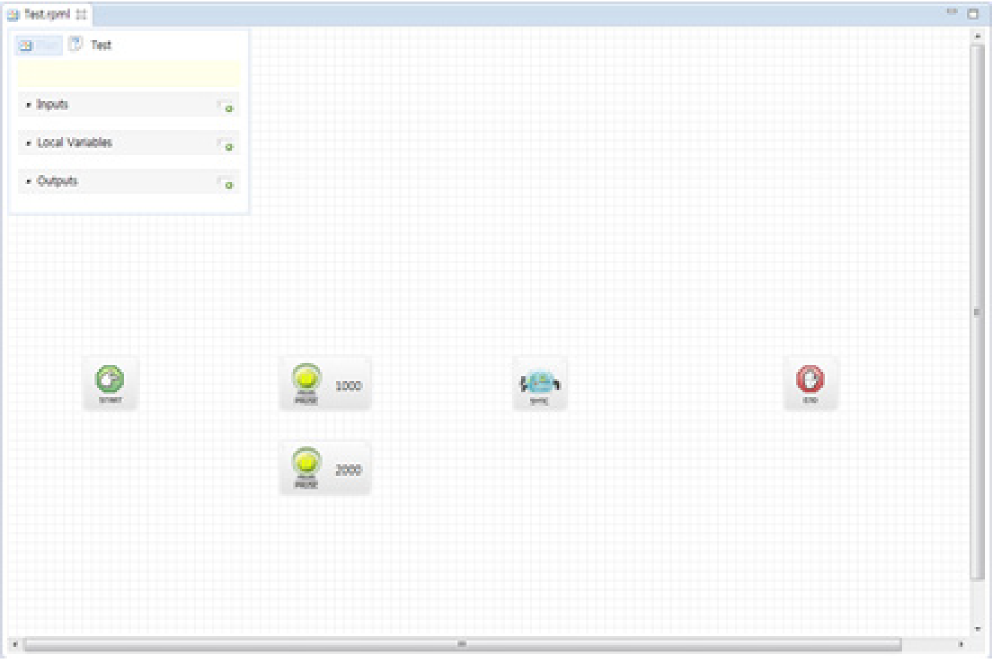
```
First condition
```

3.1.2. FLOW CONTROL 2: SYNC, PAUSE

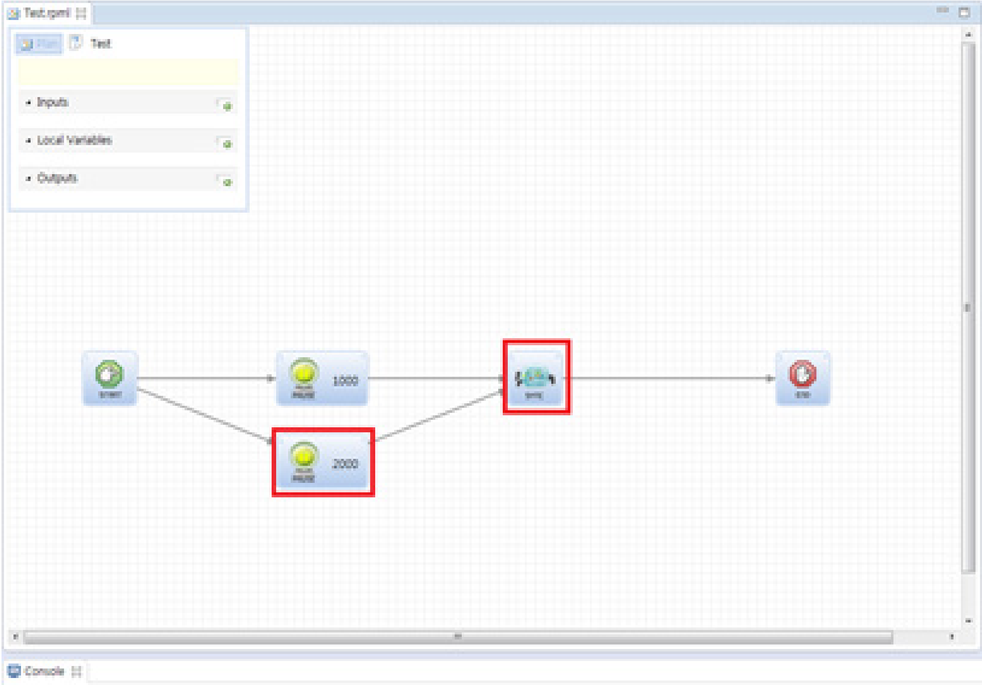
1) Adding a 'Pause' Node: Drag a 'Pause' node from the 'Component View', and drop it into the 'Editor View'. The unit of input value is millisecond (ms), and the control flow will be paused as much as you typed in the 'Pause' node.



2) Adding a 'Sync' node: Drag a 'Sync' node from the 'Component View', and drop it into the 'Editor View'.

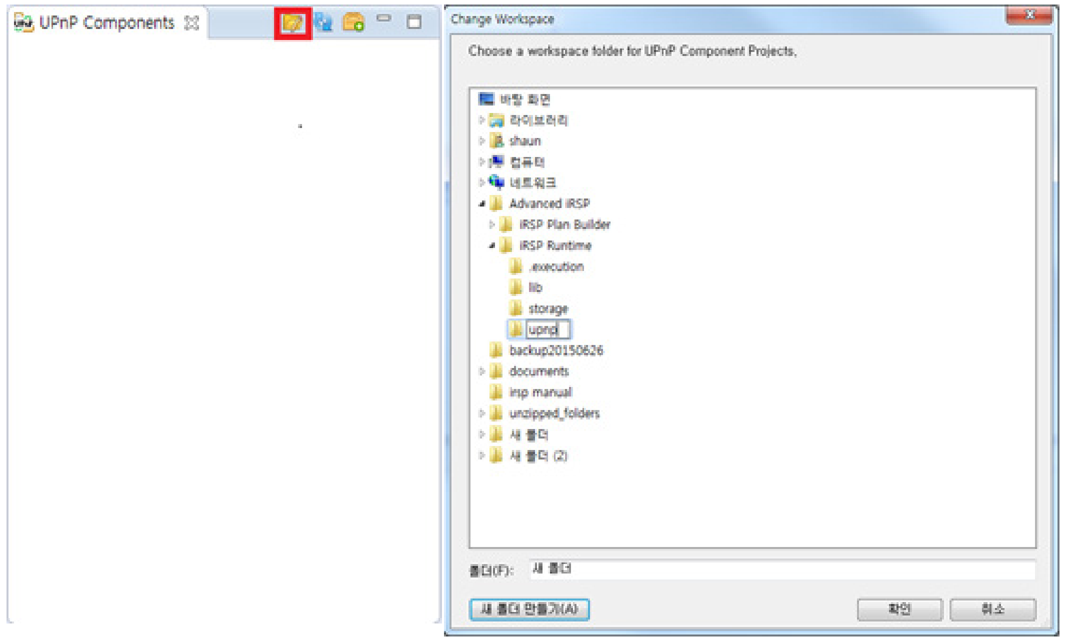


3) Only after all control flows to the 'Sync' node arrives, the control flow moves to the next node of the 'Sync' node. For this example, it shows the moment when 1000ms passed after the execution. The upper control flow executed the 1000ms waiting pause node, and the 'Sync' node is waiting for the control flow from the bottom which is running the 'Pause' node which means 2000ms waiting.



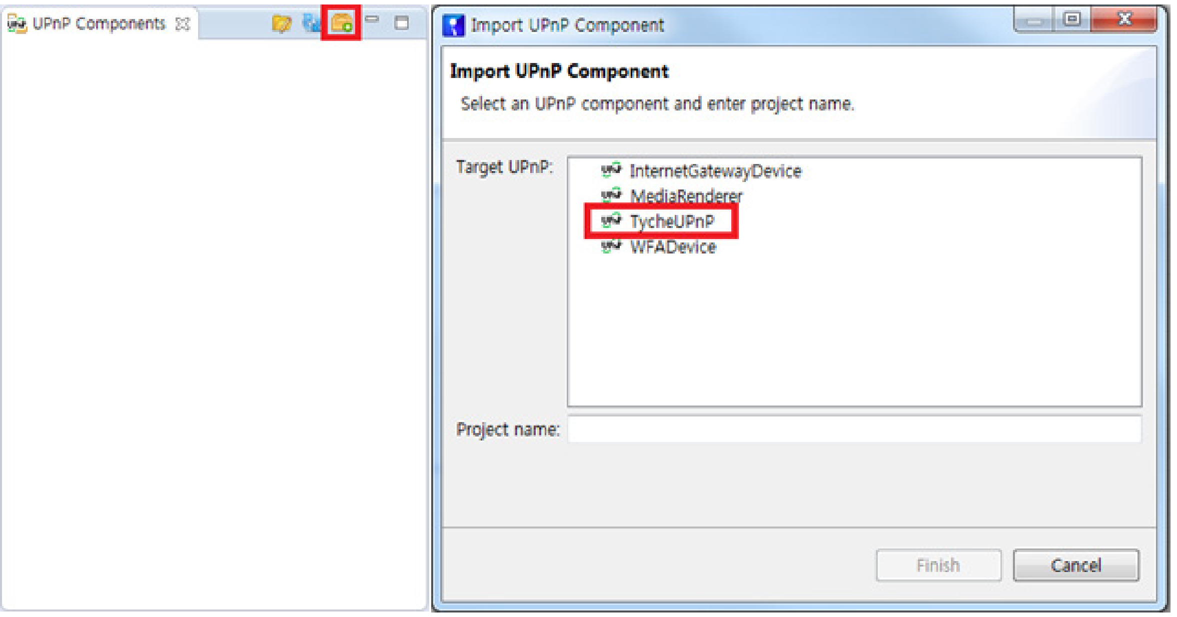
3.1.3.THE CONTROL OF UPNP COMPONENTS: UPNP ACTION, UPNP EVENT

1) Create a new folder for UPnP workspace.



Choose a workspace folder for UPnP Component Projects.

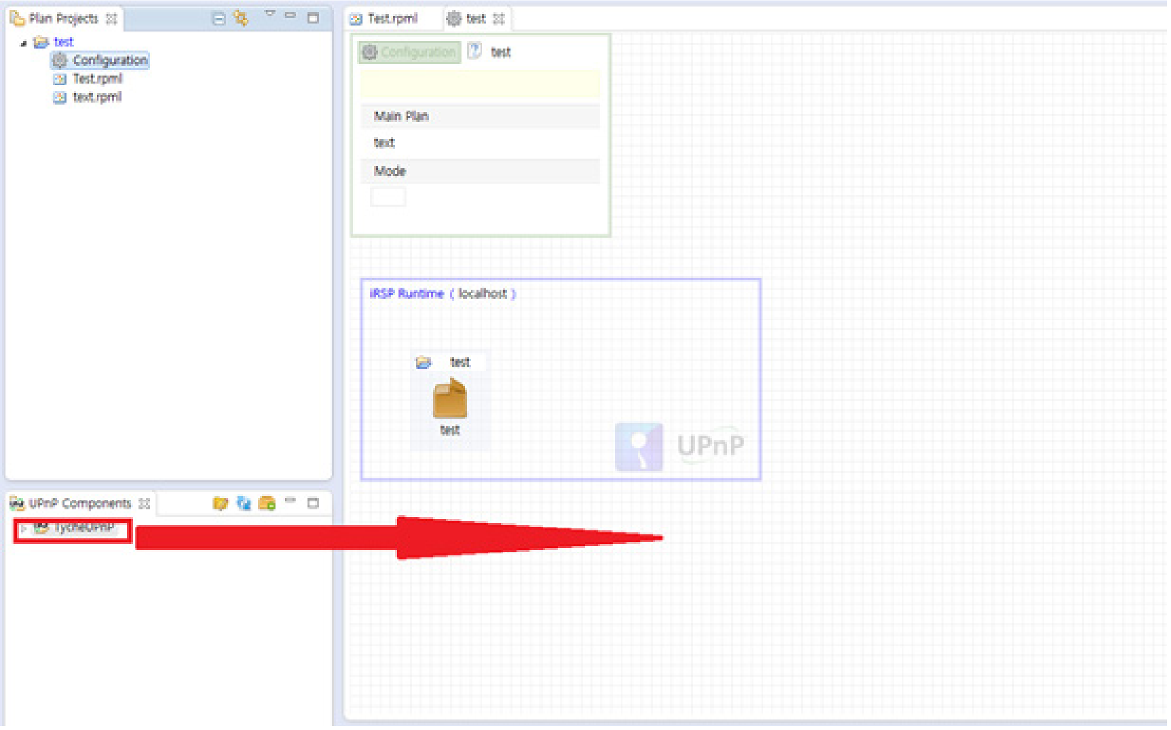
2) Search for TycheUPnP



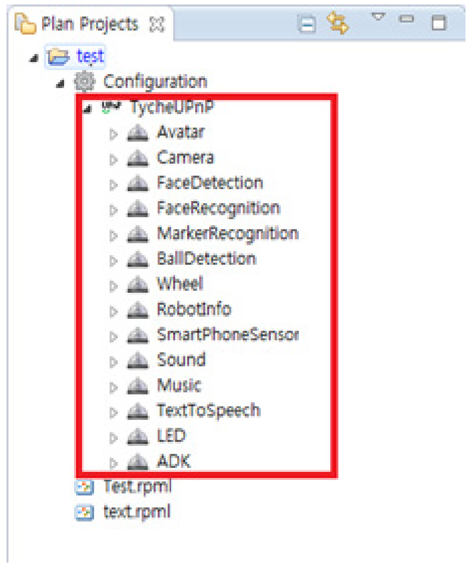
Import UPnP Component

Select an UPnP component and enter project name.

3) Add the UPnP to the Configuration file



4) Use the UPnP functions to you plan



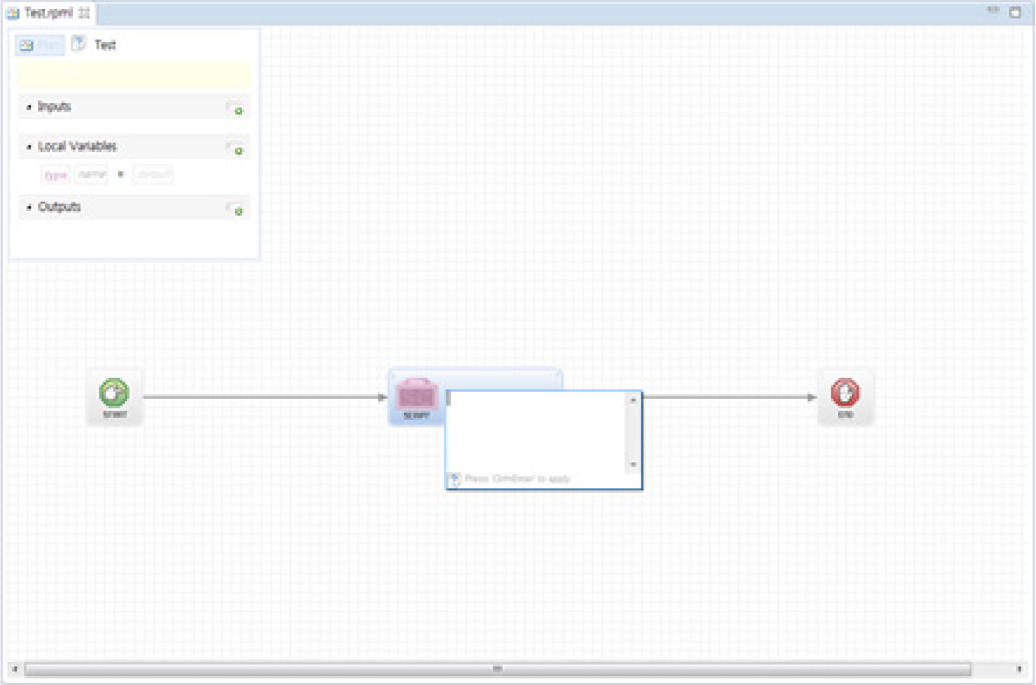
Plan Projects

- test
  - Configuration
    - TycheUPnP
      - Avatar
      - Camera
      - FaceDetection
      - FaceRecognition
      - MarkerRecognition
      - BallDetection
      - Wheel
      - RobotInfo
      - SmartPhoneSensor
      - Sound
      - Music
      - TextToSpeech
      - LED
      - ADK
  - Test.rpml
  - text.rpml

3.2. EXTENSION: SCRIPT

1) Adding an 'Script' node: Drag an 'Script' node from the 'Component View', and drop it into the 'Editor View'

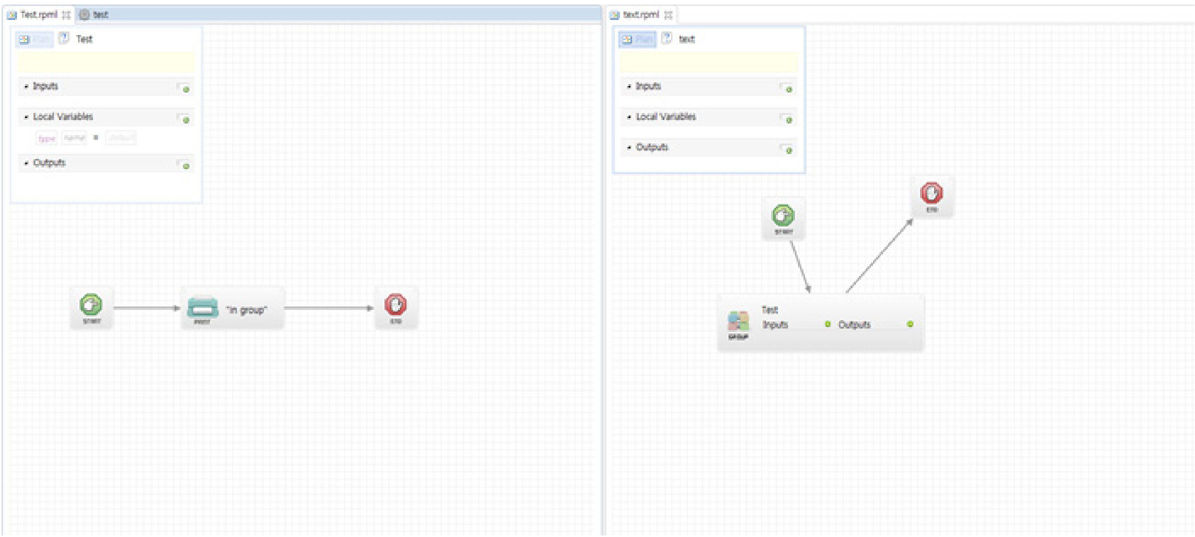
2) Double click the created 'Script' node, and the pop-up window in which you can type 'BeanShell' script will appear. Enter a script in the window. Then, Click the 'OK' button.



3.2.1. PLAN REUSE: GROUP

- iRSP supports a Group node functionality to reuse existing plans in other plans.

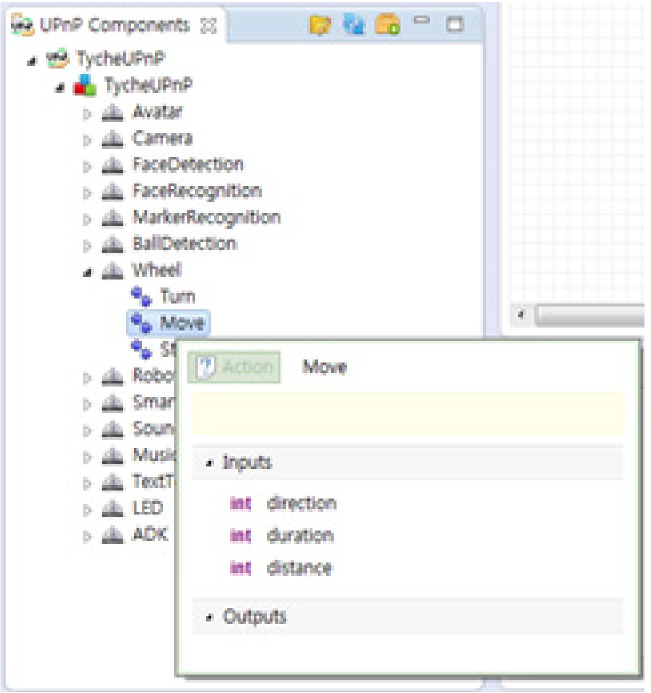
1) To use existing plans as a unit node in other plans, drag the plan node from the "Component View', and drop it into the 'Editor View'.



2) The input/output of the added group node are same as those of the existing plan, respectively.

3) Specify the variables used in the input/output of the existing plans in the input/output of the group node, respectively.

3.2.2. UPnP EVENT MONITOR



UPnP Components

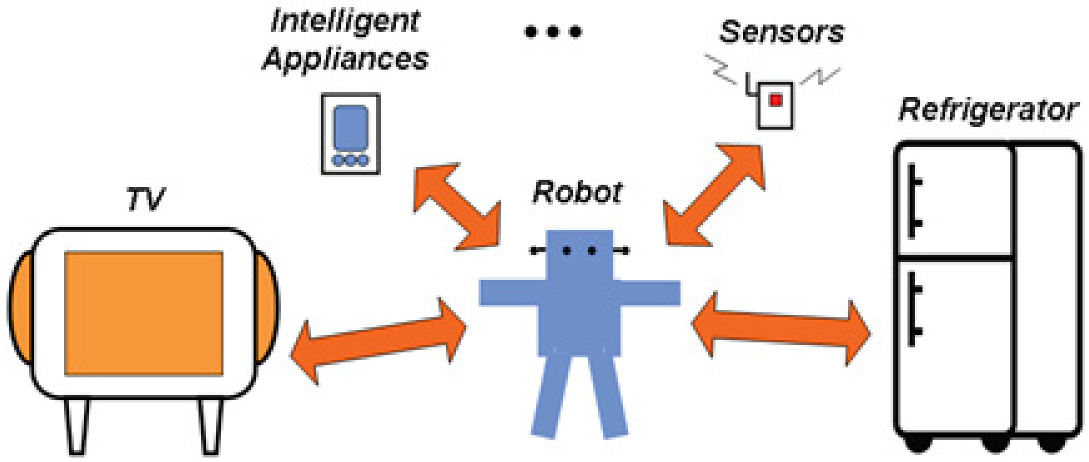
- TycheUPnP
  - TycheUPnP
    - Avatar
    - Camera
    - FaceDetection
    - FaceRecognition
    - MarkerRecognition
    - BallDetection
    - Wheel
      - Turn
      - Move
      - St
    - Robo
    - Smar
    - Soun
    - Musi
    - TextT
    - LED
    - ADK

Action Move

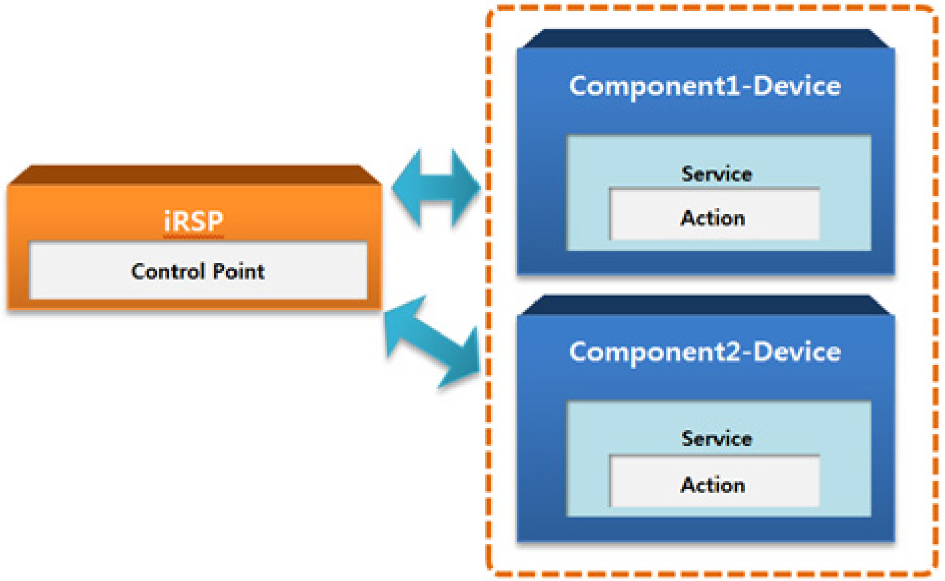
- Inputs
  - int direction
  - int duration
  - int distance
- Outputs

* You can see what inputs are required and outputs you get by hovering over a component function.

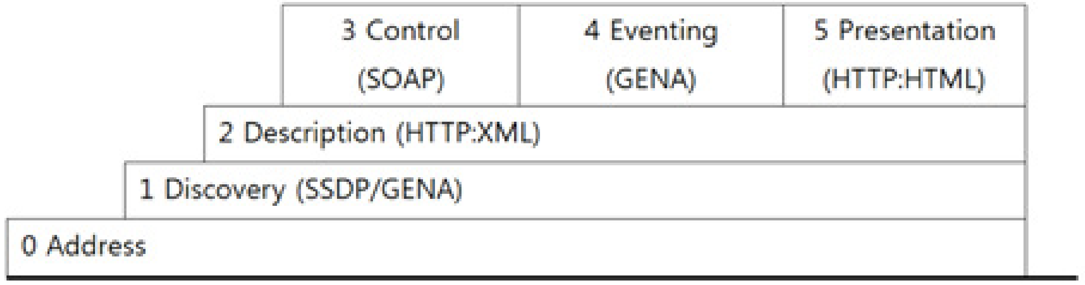
3.2.3. HOW TO USE UPnP
3.2.4. UPnP INTRODUCTION



* Universal Plug and Play (UPnP): is a standard that uses Internet and Web protocols to enable devices such as PCs, peripherals, intelligent appliances, and wireless devices to be plugged into a network and automatically know about each other.
* With UPnP, when a user plugs a device into the network, the device will announce its presence on the network to other devices.
* For instance, even though iRSP does not know the address of a device, iRSP can access it without any further steps as long as it is on the same network with iRSP.
* UPnP Structure:
A device consists of one or more services, and one service contains one or more actions. An action is what does several functions. iRSP accesses components using a 'Control Point'.

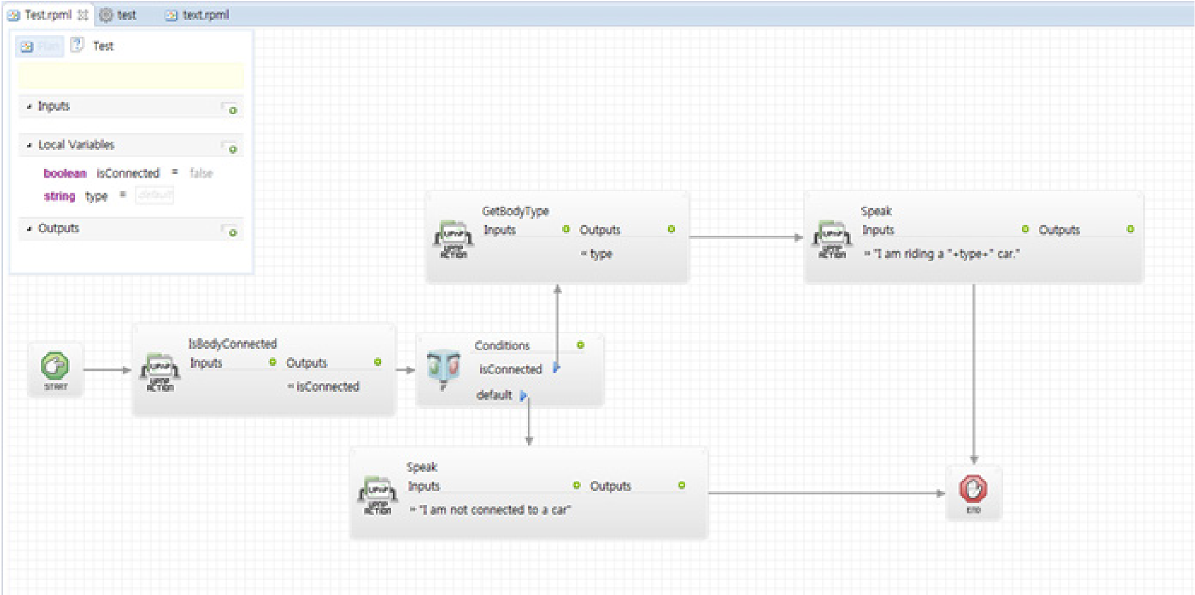


* UPnP Hierarchical Architecture:



0 Level (Address): Control point and device obtain an address to participate in the network.
1 Level (Discovery): Control point finds all devices and device advertises its availability.
2 Level (Description): Control point learns about device capabilities.
3 Level (Control): Control point invokes actions on device.
4 Level (Event notification): Control point listens to state changes of device.
5 Level (Presentation): Control point controls device and/or views device status using an HTML UI.
For more information, visit an UPnP official website (http://www.upnp.org/).

3.3. PRACTICAL USE OF IRSP
3.3.1. TYCHE WITH IRSP
3.1.2. TTS + ROBOTINFO



boolean isConnected = false
string type = default

IsBodyConnected
Inputs
Outputs
= isConnected

Conditions
isConnected
default

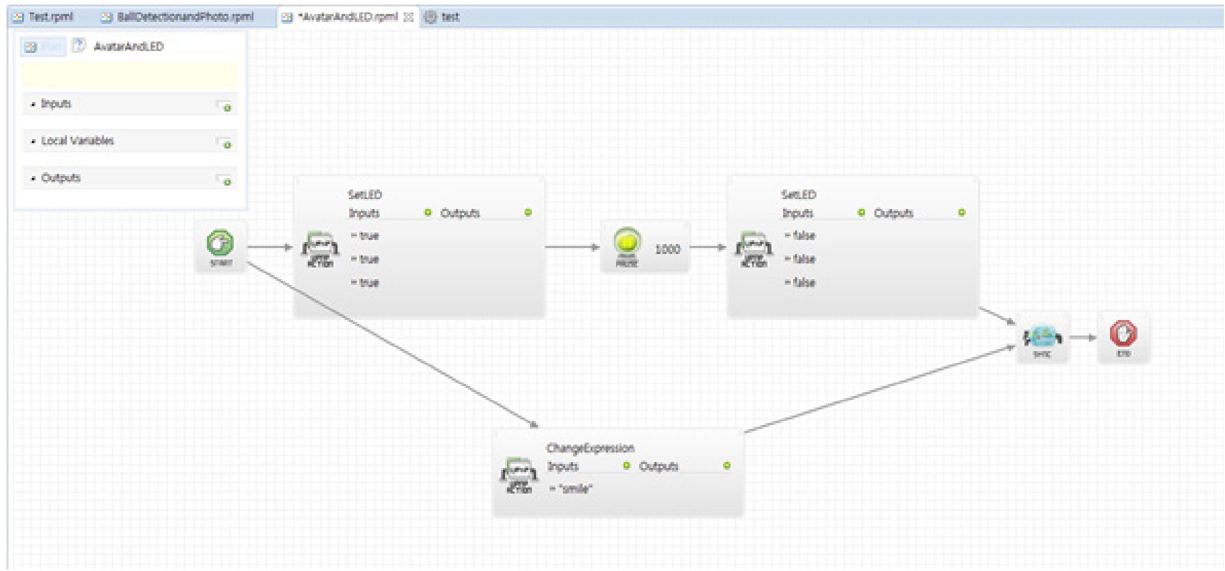
GetBodyType
Inputs
Outputs
= type

Speak
Inputs
Outputs
= "I am riding a "+type+" car."

Speak
Inputs
Outputs
= "I am not connected to a car"

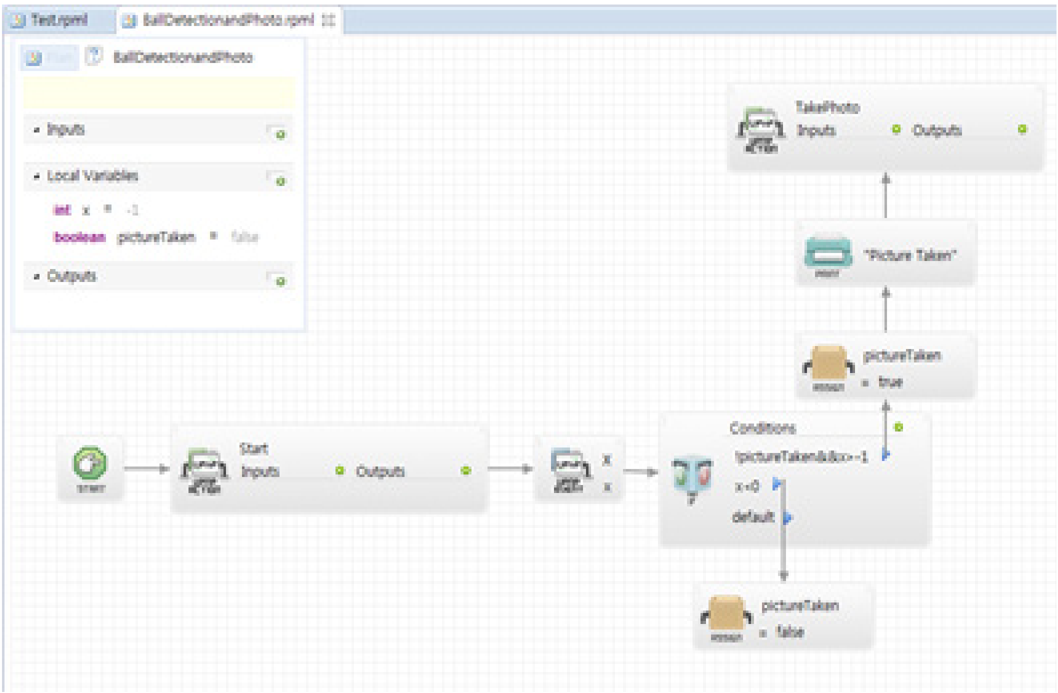
* This example will program Tyche to tell you in which type of car Tyche is, if it's in a car, and speak "I am not connected to a car" if not.

3.2.3. CHANGE EXPRESSIONS + LED



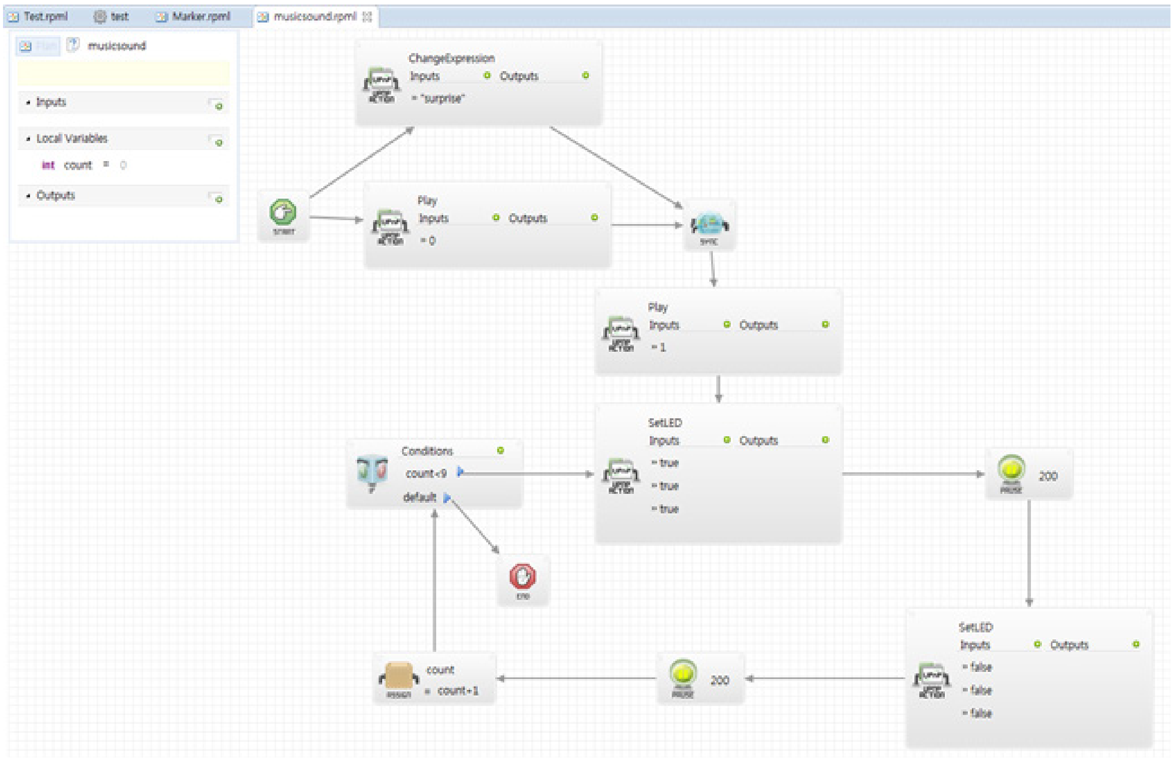
* This example will program Tyche to turn on its LED for one second and smile at the same time.

3.2.4. FACE DETECTION AND TAKE PICTURE



When Tyche detects a face, it takes a photo.

3.2.5. MUSIC + SOUND



3.3. FAQ

Q: Does iRSP support other robots beside EngKey?
A: Yes. iRSP supports various robots such as Silbot2, iRobot Create, AR Drone2, Bonavision Smartphone robot, Sphero, and Romo. Information about other robots will be offered in a component or manual update.

Q: How can I make an android UPnP Component?
A: A Service class from the UPnP Builder also can be used on the android environment. However, the code stub that creates a Device needs to be implemented in a different way.
For more detailed explanation, refer to the below Cling Library website.
http://4thline.org/projects/cling/core/manual/cling-core-manual.html#chapter.Android

Q: How can I add additional poses to the 'PoseDetection' Component?
A: The PoseDetection component is implemented using openNI Library (http://www.openni.org). To add other poses, you have to re-implement the component with the library. Currently, the 'PoseDetection' components can use only the predefined poses, and it is impossible to define additional poses. The upgraded version which allows a user to add more poses will be provided in the future update.

Q: Are there any other components for image processing?
A: Image processing components are in a developing process. As soon as they are stabilized, the components will be distributed.
For more questions, please contact us at tyche@bonavision.net .

TYCHE

The AI Companion for Kids

iRSP USER MANUAL

The Intelligent Robot Company
BONAVISION

Bonavision Inc. USA
1040 Noel Dr, #100 B & C, Menlo Park, CA 94025
Bonavision Inc. Korea
#2F JungJu Bldg., 20 Banpo-daero Seocho-Gu Seoul, Korea 137-927

www.tyche.club