# OPRoS Component Composing Tool User Manual

2011.8.29.



# **Document Information**

Item	Affiliation	Nam e	Date	Sign.
	(주)엔쓰리소프트	서동 민	2011- 08-26	
Drafter				
Fellow				
Reviewer				
QA				
Reviewer				
Approver				
Version				
Date of Issue				
Status				

# **Revision History**

Version	Revised	Revision Details	Writer	Checker
	Date			

## Table of Contents

1. <b>Overview</b>		į
1.1. Hov	v to Use This User Manual	,
1.1.1.	Purpose	,
1.1.2.	How to Use	,
1.1.3.	Appendix	,
1.1.4.	Useful Information	,
2. Component M	odeling and Overall Structure of the Composing Tool	1
2.1. Solu	ution Browser 10	1
2.1.1.	Solution Brower Structure 10	1
2.1.2.	Preferences오류! 책갈피가 정의되어 있지 않습니다.	,
2.1.3.	Solution Package13	,
2.1.4.	Component Composing Package14	;
2.1.5.	Application Component Package15	,
2.1.6.	Component Edit Pacakage 17	
2.1.7.	Repository Package	,
2.1.8.	Library Package	ł
2.1.9.	Template Package	ł
2.2. Cor	nponent Diagram Editor 20	1
2.2.1.	Layout	)
2.2.2.	Create Composite Components	ì
2.2.3.	Create Atomic Components	,
2.2.4.	Create Composite's Ports	,
2.2.5.	Create Atomic's Ports	
2.2.6.	Create Types	i
2.2.7.	Connect Connections	,
2.2.8.	Connect Note Lines	:
2.2.9.	Move Components	,
2.2.10.	Change Component Sizes	i
2.2.11.	Change Component Names	•
2.2.12.	Move Ports	,
2.2.13.	Component Popup Menu 49	ļ
2.3. Exa	mples	,
2.3.1.	An Example of Developing Composite Components	,

2.3.2.	An Example of Developing Applications	58
2.3.3.	An Example of Interworking with Component Editors	60
2.3.4.	An Example of Deploy	62
2.3.5.	An Example of Monitoring	67

## Figure List

Figure 1. Overall Structure of Composing Tools
Figure 2. Solution Browser
Figure 3. Preferences
Figure 4. Solution Package
Figure 5. Component Composing Package Popup Menu14
Figure 6. Application Component Package Popup Menu15
Figure 7. Export Application
Figure 8. Component Popup Menu16
Figure 9. Component Edit Package Popup Menu17
Figure 10. Component Edit Component Popup Menu17
Figure 11. Repository Package Popup Menu18
Figure 12. Repository Popup Menu
Figure 13. Library Package Popup Menu19
Figure 14. Template Package Popup Menu19
Figure 15. Component Diagram Editor
Figure 16. Palette
Figure 17. Diagram Edit Window
Figure 18. Create Composite Components 1
Figure 19. Create Composite Components 2
Figure 20. Create Atomic Components 1
Figure 21. Create Atomic Components 2
Figure 22. Create Atomic Components 3
Figure 23. Create Atomic Components 4
Figure 24. Create Atomic Components 5
Figure 25. Create Composite's Ports 1
Figure 26. Create Composite's Ports 2
Figure 27. Create Composite's Ports 3
Figure 28. Create Composite's Ports 4
Figure 29. Create Atomic's Ports 1
Figure 30. Create Atomic's Ports 2
Figure 31. Service Port Dialog
Figure 32. Data Port Dialog
Figure 33. Event Port Dialog
Figure 34. Create Atomic's Ports 3

Figure 35. Create Atomic's Ports 4	5
Figure 36. New Type or Import Dialog	6
Figure 37. Add Method Type 1	7
Figure 38. Add Method Type 2	7
Figure 39. Add Method Type 3	8
Figure 40. Add Method 1	9
Figure 41. Add Method 2	9
Figure 42. Add Method Parameter 1 40	0
Figure 43. Add Method Parameter 2 40	0
Figure 44. Completion of New Type or Import	1
Figure 45. Type Select	1
Figure 46. Connect Connections 1	2
Figure 47. Connect Connections 2	2
Figure 48. Connect Connections 3 43	3
Figure 49. Connect Note Lines 1 44	4
Figure 50. Connect Note Lines 2 44	4
Figure 51. Move Components 1 45	5
Figure 52. Move Components 2	5
Figure 53. Change Component Sizes 1 46	6
Figure 54. Change Component Sizes 2 46	6
Figure 55. Change Component Names 1 47	7
Figure 56. Change Component Names 2 47	7
Figure 57. Move Ports 1 48	8
Figure 58. Move Ports 2 48	8
Figure 59. Component Popup Menu 49	9
Figure 60. Open Property 50	0
Figure 61. Open Monitoring Dialog	1
Figure 62. Example of Developing Composite Components 1	2
Figure 63. Example of Developing Composite Components 2	2
Figure 64. Example of Developing Composite Components 3	3
Figure 65. Example of Developing Composite Components 4	3
Figure 66. Example of Developing Composite Components 5	4
Figure 67. Example of Developing Composite Components 6	4
Figure 68. Example of Developing Composite Components 7	5
Figure 69. Example of Developing Composite Components 8	6
Figure 70. Example of Developing Composite Components 9	7

Figure 71. Example of Developing Applications 1	58
Figure 72. Example of Developing Applications 2	58
Figure 73. Example of Developing Applications 3	59
Figure 74. Example of Interworking with Component Editors 1	60
Figure 75. Example of Interworking with Component Editors 2	60
Figure 76. Example of Interworking with Component Editors 3	61
Figure 77. Example of Interworking with Component Editors 4	61
Figure 78. Example of Deploy 1	62
Figure 79. Example of Deploy 2	62
Figure 80. Example of Deploy 3	63
Figure 81. Example of Deploy 4	63
Figure 82. Example of Deploy 5	64
Figure 83. Example of Deploy 6	64
Figure 84. Example of Deploy 7	65
Figure 85. Example of Deploy 8	65
Figure 86. Example of Deploy 9	66
Figure 87. Example of Monitoring 1	67
Figure 88. Example of Monitoring 2	68
Figure 89. Example of Monitoring 3	69
Figure 90. Example of Monitoring 4	69
Figure 91. Example of Monitoring 5	70
Figure 92. Example of Monitoring 6	70
Figure 93. Example of Monitoring 7	71

### 1. Overview

The robot component composer tool has functions to fetch the developed OPRoS components, assemble them, develop composite components and applications, and distribute the created applications.

#### 1.1. How to Use This User Manual

This manual is written to follow easily along with the OPRoS job training, and materials for beginners of the OPRoS.

This user manual describes each item of the component composer tool in detail, adds figures to every item, and you can find a page number of the corresponding item through the table of contents.

#### 1.1.1. Purpose

This manual explains easy to follow how to install the component composer tool, how to use it basically, and how to manage it.

#### 1.1.2. How to Use

If you have already installed the Eclipse, refer from Chapter 3.

#### 1.1.3. Appendix

The appendix simply explains terms easily confusable in this document.

#### 1.1.4. Useful Information

Easy-to-use methods are provided with a mark of "TIPS: " for each item.

2. Component Modeling and Overall Structure of the Composing Tool



Figure 1. Overall Structure of Composing Tools

It is comprised of the solution browser, which models application and composite components and manages library folders, the description window, which can see description for each component, the component diagram editor, which can model composite components, the component diagram editor tree window, which can identify model elements of a component diagram, the console window, which shows the progress of a project's source, and the task view that shows the progress of modeling.

#### 2.1. Solution Browser

2.1.1. Solution Browser Structure



#### Figure 2. Solution Browser

When executing the tool, it shows the solution browser as the above structure.

- Solution Package: Saves or opens projects, and creates new projects.
- Component Composing Package: Adds new application packages and diagrams.
- Application Component Package: Models application components on the main diagram.
- Component Edit Package: Creates developable or editable components.
- Robot Package: Manages robots and each node of robots.
- Repository Package: Accesses to a remote server computer to fetch libraries

stored in the server, and shows them.

- Library Package: Fetches information in folders specified as the library folder in preferences, and shows them.
- Template Package: Fetches samples of applications or components, folders specified as the template folder in preferences, and shows them.

#### 2.1.2. Preferences

a. Select 'Window->Preferences' on the top menu of Eclipse to open the preferences window.

<u>File Edit N</u> avigate Se <u>a</u> rch <u>P</u> roject T	PL <u>R</u> un <u>T</u> oolRun	Window Help	
		<u>N</u> ew Window New <u>E</u> ditor	
		<u>O</u> pen Perspective Show <u>V</u> iew	
€ Preferences		Customize Perspective Save Perspective <u>A</u> s <u>Reset Perspective</u> <u>Close Perspective</u>	
type filter text	OPRoS_CC_Dire	Close All Perspectives	
⊞⊸ General ⊞⊸ Ant	Folder Setting		
E C/C++	Local Repository	Click	<u>Browse</u>
install/Update	Template Path:	Company	<u>B</u> rowse
	<ul> <li>Deploy Option for</li> <li>Release</li> </ul>	Component :	
Component Composer OPRoS_CC Compile OPRoS_CC Copyright OPROS_CC_Directory Component Editor Plug-In Development Provisioning Admin Remote Systems Report Design Run/Debug Tasks Team Usage Data Collector Validation XML	O Debug	(	Restore <u>D</u> efaults) <u>Apply</u>
0			OK Cancel

Figure 3. Preferences

- b. When specifying a physical path of a library in the preferences window, the specified path is reflected into the solution browser's library package area to be shown.
- c. Specify the path to the corresponding program's executable files for the executable file path.

#### 2.1.3. Solution Package



#### Figure 4. Solution Package

- New Project: Terminates the existing project, and creates a new project.
- Save Project: Stores the project.
- Open Project: Fetches a project.

2.1.4. Component Composing Package



Figure 5. Component Composing Package Popup Menu

Add Application: Creates a new application package and a dialog.

#### 2.1.5. Application Component Package

a. Package Popup Menu





- Delete: Deletes the selected package.
- Deploy: Deploys the selected package.

🔁 Solution Browser 🕱 🛛 🏠 🗇 🗘 🗖 🗖	Select Folder	? 🛛
Solution Component Composing Application Export Application Deploy Deploy Deploy(tar)	<ul> <li>☞ 바탕 화면</li> <li>☞ 나 방문서</li> <li>☞ ☞ mycom</li> <li>☞ ☞ 내 네트워크 환경</li> <li>☞ ☞ 페꼬리</li> <li>☞ 합포저 메뉴얼</li> </ul>	
	폴더: 내 문서	
	새 폴더 만들기( <u>M</u> ) 확인 취:	<u>ک</u> , با

Figure 7. Export Application

Export Application: Able to distribute the selected application after choosing the specified folder.

b. Component Popup Menu



Figure 8. Component Popup Menu

- Delete: Deletes the selected component.
- Export Component: In the case of a composite component, it is exported as a XML form to the outside.

#### 2.1.6. Component Edit Package

a. Package Popup Menu





- All Synchronization: Synchronizes the composer and editor component information.
- b. Component Popup Menu





- Delete: Deletes the selected component.
- Synchronization: Synchronizes the selected component with component information on the editor.
- Open Component Edit: Creates a project with the selected component on the component editor, and changes the tool into the component editor.

2.1.7. Repository Package



Figure 11. Repository Package Popup Menu

Connect Repository: Enters information on server computer's IP and ports to add a repository server.



#### Figure 12. Repository Popup Menu

- Connect: Connects to a repository server to read a library, and shows it.
- Delete: Deletes the repository server.

2.1.8. Library Package



Figure 13. Library Package Popup Menu

- Import Component: It can fetch atomic and composite components in other folders to the selected folder.
- Refresh: If content of the folder storing components is changed, it reads new information on the changed folder and reflects it.
- Preferences: Refer to the preferences
- 2.1.9. Template Package



#### Figure 14. Template Package Popup Menu

Load Template: Fetches template components from the path specified in 'Preferences – Template Path.'

#### 2.2. Component Diagram Editor

#### 2.2.1.Layout



Figure 15. Component Diagram Editor

- Palette: It is a tool that collects components, ports and relations available to use on the diagram, which users can select components and relations to be created from the palette to create them on the diagram edit window.
- Diagram Edit Window: It is an edit window that users can model application and composite components by adding objects in the palette or solution browser.

a. Palette



#### Figure 16. Palette

- Select: Used when selecting a single object on the diagram edit window.
- Marquee: Used when selecting multiple objects on the diagram edit window.
- Composite: Used when creating a composite component on the diagram edit window.
- Atomic: Used when creating an atomic component on the diagram edit window.
- Service Required: Used when adding a 'Service Required' to the selected component on the diagram edit window.
- Service Provided: Used when adding a 'Service Provided' to the selected component on the diagram edit window.
- Data OutputPort: Used when adding a 'Data OutputPort' required to the selected

component on the diagram edit window.

- Data InputPort: Used when adding a 'Data InputPort' to the selected component on the diagram edit window.
- Event OutputPort: Used when adding an 'Event OutputPort' to the selected component on the diagram edit window.
- Event InputPort: Used when adding an 'Event InputPort' to the selected component on the diagram edit window.
- Connection: Used when connecting a component's port.
- Note: Used when adding a note.
- Note Link: Used when connecting objects (components, ports) related to a note.



b. Diagram Edit Window

Figure 17. Diagram Edit Window

#### 2.2.2. Create Composite Components



a. Select the Composite from the palette to create a new composite component.

b. Click the location to draw the component on the diagram edit window.



c. A basic component is created on the diagram edit window as below.

Figure 19. Create Composite Components 2

#### 2.2.3. Create Atomic Components

- 0 🖽 Application diagram 🔀 😳 Palette D Select 🖡 门 Marquee Components -📄 Composite 🚔 Atomja Servi Click Require C Service Provided 🗾 Data OutputPort 🔽 Data InputPort Event OutputPort Sevent InputPort 🔨 Connection B₀ Notes 0 Note 🔨 Note Link
- a. Select the Atomic from the palette to create a new atomic component.

Figure 20. Create Atomic Components 1

- b. Click the location to draw the atomic component on the diagram edit window.
- c. When the 'New OPRoS Component' window is appeared to ask about information on the atomic component to be created, enter input items and click OK.

Component Name		
Selecting Language	MSVC C++	~
ОК	Cancel	

#### Figure 21. Create Atomic Components 2

- Component Name: Name of the component to be created
- Selecting Language: Select a development language between MinGW
   C++ and MSVC C++



d. A basic atomic component is created on the diagram edit window as below.

Figure 22. Create Atomic Components 3



e. In a different way, select an atomic component from the solution browser.

Figure 23. Creates Atomic Components 4

f. Drag and drop it to the diagram edit window while clicking the mouse.



Figure 24. Create Atomic Components 5

g. When releasing the mouse button, an atomic component is drawn on the diagram edit window.

#### 2.2.4. Create Composite's Ports

- 0 🖽 \*Application diagram 🖂 🚼 Palette D Select , []] Marquee 🕄 Components 00 Composite 🛗 Atomic 3 Service Required C Servid Click Provided Component0 🔼 Data OutputPort 🔽 Data InputPort Event OutputPort ♦ Event InputPort 🔨 Connection B₀ Notes  $\Leftrightarrow$ 📄 Note 🔨 Note Link
- a. Select a port to be added from the palette.

Figure 25. Create Composite's Ports 1

b. Click the component, which you want to add a port, on the diagram edit window.



Figure 26. Create Composite's Ports 2



c. The corresponding port is added on the top left of the component.

Figure 27. Create Composite's Ports 3



d. In a different way, a port can be added by drag-and-drop as adding an atomic component.

Figure 28. Create Composite's Ports 4

#### 2.2.5. Create Atomic's Ports



a. Select a port to be added from the palette.

Figure 29. Create Atomic's Ports 1

b. Click the component, which you want to add a port, on the diagram edit window.



Figure 30. Create Atomic's Ports 2

- c. For the Atomic, unlike the Composite, a dialog window is appeared to enter a name/properties etc. of the port.
  - A. Service Port

Vame : Jsage :	 required	Тура	e Name : e Reference	NotReference	~
Description	:				
Paulaa Da	rt Type List :		New	Type or Import	
service Po					
Service Po					
Service Po					
Service PO					
Service PO					

#### Figure 31. Service Port Dialog

- Name: Enter a name of the port.
- Type Name: Specify a type.

- Description: Enter description about the port.
- New Type or Import: Add a new type (refer to the Create Types.)
- Service Port Type List: Select a type created.
- B. Data Port

UPRoS Da Input Data Por	ata Port Dialog t			2
Name :		DataType :		~
Usage :	input	Reference :		]
Policy :	FIFO 💌	QueueSize :		1
Description :			< Clear Reference	>
Data Type Fil	e List ;		ewrype or import.	<u> </u>

#### Figure 32. Data Port Dialog

- Name: Enter a name of the port.
- Data Type: Specify a type.
- Policy: Specify a policy.
- Queue Size: Specify a queue size.
- Description: Enter description about the port.
- Clear Reference: Initialize data types and references.
- New Type or Import: Add a new type (refer to the Create Types.)
- Data Type File List: Select a type created.

C. Event Port

11 12 13 13 13 13 13 13 13 13 13 13 13 13 13	
DataType :	*
Description :	
to modify or regenerate so	urce
	Data Lype : Description : to modify or regenerate so

Figure 33. Event Port Dialog

- Name: Enter a name of the port.
- Data Type: Specify a type.
- Description: Enter description about the port.

d. When pressing the OK button to close the dialog, the corresponding port is added on the top left of the component.



Figure 34. Create Atomic's Ports 3



e. A port can be added by drag-and-drop as adding the Composite's ports.

Figure 35. Create Atomic's Ports 4

#### 2.2.6. Create Types

a. Press the New Type or Import button to open the Create window.

🖨 Define ServiceType	×
OPRoS Service Types	
service_port_type_profile	
	-
Service File Name :	
	20
Import OK Cancel	

#### Figure 36. New Type or Import Dialog

- Service File Name: A name of the xml file to store the created type.
- Import: Fetches the type xml file from the outside.

b. Click the right mouse button on the Service\_port\_type\_profile to open the popup menu.

Correction Service Type OPRoS Service Types		×
- UPHOS Service Types	Add Method Type Delete	
Service File Name :	OK Cancel	

Figure 37. Add Method Type 1

c. Select the Add Method Type, and enter a type name.



Figure 38. Add Method Type 2

 d. When pressing the tree expansion button on the left of the Service\_port\_type\_profile, you can see the created type.

🖨 Define ServiceType	×
OPRoS Service Types	3
pervice_port_type_profile     test	
test.	
Service File Name :	
Import OK Cancel	)

Figure 39. Add Method Type 3

e. After selecting the created type and opening the popup menu, select the Add Method to open a window to create methods.

🛢 Define S	erviceType		
OPRoS Service.	ce Types port_type_profile Add Method Delete		
Service File	Name :	ок (	Cancel

Figure 40. Add Method 1

f. Enter the Service Name, Return Data Type and Call Type, and create the method.

Service Name :		
Return Data Type :	void	~
Call Type :	blocking	~

#### Figure 41. Add Method 2

- Service Name: Enter a name of the method to create.
- Return Data Type:
- Call Type: Enter description about the port.

g. After selecting the created method, select the Add Method Parameter to open the Create Parameter window.

🖨 Define ServiceType	$\mathbf{X}$
OPRoS Service Types	
Lesity void-blocking	Add Method Parameter Delete
Service File Name :	
Import O	K Cancel

Figure 42. Add Method Parameter 1

h. Enter the Parameter Name and Data Type to create the parameter.



Figure 43. Add Method Parameter 2

i. When entering the created type name into the Service File Name and clicking the OK button, it is completed.

PRoS Servic	e Types —		
🖃 service_p	ort_type_prof	ile	
🖻 test			
E te:	st():Void:Dioci	king	
	lestint		
ervice File N	lame : test		
	anno - looq		

Figure 44. Completion of New Type or Import

j. When selecting the created type from the Service Port Type List and the type name from the Type Name, it can be used.

OPRoS Se Required Serv	rvice Port Di ice Port	alog					$\mathbf{X}$
Name : Usage : Description :	required		Type Na Type Re	me : ference	test test, An	m	
asas,xml test,xml Click He	nype List :			14640	туре от	mport	
If do addition,	you need to mo	odify or r	egenerat	e source	9		
			C	ОK		Canc	el

Figure 45. Type Select

#### 2.2.7. Connect Connections

a. Select the Connection from the palette.



Figure 46. Connect Connections 1

 After selecting a port of the source component on the diagram edit window, move the mouse to select a port of the target component.



Figure 47. Connect Connections 2



c. Two ports are connected with a line.

Figure 48. Connect Connections 3

*	Lines	must	be	attached	with	the	direction	of	connections	as	below.
		111001	20	anaonoa			anootion	0.	00111000110110	ao	0010111



#### 2.2.8. Connect Note Lines

- 8 🖽 \*Application diagram 🔀 😳 Palette D Select , []] Marquee Components 0 📄 Composite Label 🛗 Atomic Service Required C Service Provided <<atomic>> 🗖 Data OutputPort E1 🔽 Data InputPort Event OutputPort Event InputPort 🔨 Connection 🗈 🛛 Notes Note Click
- a. Select the Note from the palette to create it on the diagram edit window.

#### Figure 49. Connect Note Lines 1

 After selecting the NoteLine, click the Note and a component (or a port) to connect them (the NoteLine is not distinguished between sources and targets unlike the Connection.)



Figure 50. Connect Note Lines 2

#### 2.2.9. Move Components



a. When dragging a component after selecting it by the mouse, it is moved.

Figure 52. Move Components 2

b. If using the 'Alt + arrow keys' after selecting a component, it can be moved minutely.

#### 2.2.10. Change Component Sizes

a. If dragging the handle bound of a component while clicking it, the component size can be changed.









- -

#### 2.2.11. Change Component Names

a. If double-clicking the name area of a component on the diagram edit window, a text edit window is created, and the name can be modified.



Figure 55. Change Component Names 1

b. After selecting a component, the name can be modified through the name property on the properties window.

		l ~
Property	Value	-
🖃 Component Info		
ID	4f6e03,12ba94d	
Name	Component0	
Version	1,0	-
Execution CPU Info		
Exe_env_cpu		
Execution OS Info		
Exe_env_os		
Exe_env_os_vers	le .	~

Figure 56. Change Component Names 2

#### 2.2.12. Move Ports



a. When dragging a port, which you want to move, after selecting it, the port will be moved along the component's border.

Figure 58. Move Ports 2

#### 2.2.13. Component Popup Menu



Figure 59. Component Popup Menu

Delete: Deletes a component selected on the diagram edit window.



• Open Property: Sets properties and monitoring variables of a component.

#### Figure 60. Open Property

- Properties: Adds/Deletes variables of a component.
- Monitoring Variable: Adds/Deletes the monitoring variables.

 Open Monitoring Dialog: Able to check changes of the monitoring variable's values.

ariable				
heck	name	type	value	
)	1	string	TEST	
	2	string	TEST2	
	3	string	TEST3	
ariable Log				
2				
	]			

Figure 61. Open Monitoring Dialog

- Run Monitoring: Monitors the selected component.
- Stop Monitoring: Stops to monitor the selected component.
- Open ComponentEdtior: Popup menu applying only to atomic components, which calls a component editor to the corresponding component.
- Z-Order: Moves the selected component back and forth on the z-axis.
  - Bring Forward: Moves the selected component forward on the z-axis.
  - Send Back: Moves the selected component backward on the z-axis.

#### 2.3. Examples

- 2.3.1. An Example of Developing Composite Components
  - a. Create a composite component below the package.



Figure 62. Example of Developing Composite Components 1

b. Open the internal diagram for editing the component (double-click the component's internal diagram in the solution browser, or use the Open Diagram from the popup menu of the component on the diagram edit window)



Figure 63. Example of Developing Composite Components 2



c. The internal diagram of the default component created is opened.

Figure 64. Example of Developing Composite Components 3

d. Drag-and-drop components in the Library or Repository to add them into the diagram edit window.



Figure 65. Example of Developing Composite Components 4

e. Connect the related ports each other using the connection. If necessary, add ports to the composite component, and connect them with the internal component.



Figure 66. Example of Developing Composite Components 5

f. When adding a port to the composite component in the internal diagram, the port is added to be represented also on the reduced component as below.



Figure 67. Example of Developing Composite Components 6

g. Execute the component's 'Popup Menu>Export Component' to save the component as a XML file.



Figure 68. Example of Developing Composite Components 7

 In the solution browser's library, the released component can be added to the component repository through the 'Popup Menu>Import Component' (\* Select a folder storing the released component files separately because all the lower folders are searched.)

🛅 Solution Browser 🕱 🛛 🟠 🗇 😅 🗖	
<ul> <li>Solution</li> <li>Component Composing</li> <li>Application</li> <li>Component Editing</li> <li>Robots</li> <li>Remote Repository</li> <li>Remote Repository</li> <li>Hellot</li> <li>Hellot</li> <li>Mess</li> <li>Refresh</li> </ul>	
폴더 찾아보기	? 🗙
[ 바탕 화면	
폴더: 내 문서	
새 폴더 만들기( <u>M</u> ) 확인	취소 ::

Figure 69. Example of Developing Composite Components 8

i. It can be seen that the component is added to the component repository.



Figure 70. Example of Developing Composite Components 9

#### 2.3.2. An Example of Developing Applications

a. Double-click the Application diagram in the Application package to open it.



Figure 71. Example of Developing Applications 1

b. Add components to the diagram edit window. The atomic components in the library or repository can be added. Or composite components made in the application package can be added directly or by importing them to the library.



Figure 72. Example of Developing Applications 2

c. Applications can be released by executing the 'Popup Menu>Export Application' of the application package. Specify a location to be released and press the OK.



Figure 73. Example of Developing Applications 3

#### 2.3.3. An Example of Interworking with Component Editors

c. After selecting the Atomic Component on the palette, drag it to the diagram or click a desired location.



Figure 74. Example of Interworking with Component Editors 1

d. When clicking the OK after selecting a name of the component to be created and a development language (MinGW C++/ MS VC C++,) the component is created.

Component Name	1	
Selecting Language	MSVC C++	~
ОК	Cancel	

Figure 75. Example of Interworking with Component Editors 2



e. After adding ports required for the component, click the 'Open Component Edit' on the Atomic Component popup menu.



Figure 77. Example of Interworking with Component Editors 4

f. You can see that it is changed into the component editor as a component editor project for the corresponding component is created (refer to the component editor manual for how to use the editor.)

#### 2.3.4. An Example of Deploy

g. Execute the Robots' 'Popup Menu>Add Robot' to add a new robot.

🛅 Solution Browser 🕱 🛛 🏠 🗇 👄	
<ul> <li>Solution</li> <li>Component Composing</li> <li>Application</li> <li>Application diagram</li> <li><a tornic="">&gt; test00</a></li> <li>Component Editing</li> <li>Robots</li> <li>Remo</li> <li>Add Robot</li> <li>Local Repository</li> </ul>	
🖨 Add NodePackage	X
Name	
	OK Cancel

Figure 78. Example of Deploy 1

h. Execute the 'Popup Menu>Add Node' of the added robot to add a new node.



Figure 79. Example of Deploy 2

i. Enter IP and Port information into the node.



Figure 80. Example of Deploy 3

j. Drag-and-drop the node to add it into the component.



Figure 81. Example of Deploy 4

k. Execute the Application Package's 'Popup Menu>Deploy' to deploy the component into the node. You can execute one node or all the nodes at a time.



Figure 82. Example of Deploy 5

I. The released result can be checked on the Console window.

🖳 Task View 🛛 📮 Console	~ - 8
♦ 서비 연결 ip[127,0,0,1] port[3001]	
🔗 파일 전송: Application,xml size[1931]	
🧳 서버 응답: Application.xml size[0] : ok	
🥏 파일 전송: Application₩Component1,xml size[494]	
🧳 서버 응답: Application₩Component1,xml size[0] : ok	
🔷 파일 전송: Application₩Component1_view,xml size[336]	
🕢 서버 응답: Application₩Component1_view,xml size[0] : ok	
🧼 파일 전송 완료	

Figure 83. Example of Deploy 6

m. Execute the node's 'Popup Menu>Get Application List' to fetch the application list from the repository of the computer connected with the node, and indicate it.



Figure 84. Example of Deploy 7

 If succeeded in fetching the application list, an application package is created below the node, and the server's response information is printed on the console window. If failed, a message of 'Connection is Failed' is appeared.



Figure 85. Example of Deploy 8

o. Application List Menu



Figure 86. Example of Deploy 9

- Run Application: Executes the stopped application.
- Stop Application: Stops an application in execution.
- Get Application State: Prints state of the application in execution on the console window.
- Get Component List: Fetches a list of components included in the application to show it.

#### 2.3.5. An Example of Monitoring

p. Execute the Robots' 'Popup Menu>Add Robot' to add a new robot.

🛅 Solution Browser 🕱 🛛 🏠 🗇 🔿	- 0
<ul> <li>Solution</li> <li>Component Composing</li> <li>Application</li> <li>Application diagram</li> <li><atomic>&gt;test00</atomic></li> <li>Component Editing</li> <li>Remo</li> <li>Add Robot</li> <li>Local Hepository</li> </ul>	
🖨 Add NodePackage	X
Name	
	OK Cancel

Figure 87. Example of Monitoring 1

q. Execute the added robot's 'Popup Menu>Add Node' to add a new node.



Figure 88. Example of Monitoring 2

r. Enter IP and Port information into the node.



Figure 89. Example of Monitoring 3

s. Drag-and-drop the node to add it into the component.



Figure 90. Example of Monitoring 4

t. If clicking the Run Monitoring on the popup window of the component to be monitored, the corresponding component can be monitored.



Figure 91. Example of Monitoring 5



u. It is executed as below.

Figure 92. Example of Monitoring 6



v. If pressing the 'Popup Menu>>Stop Monitoring,' the monitoring can be stopped.

Figure 93. Example of Monitoring 7