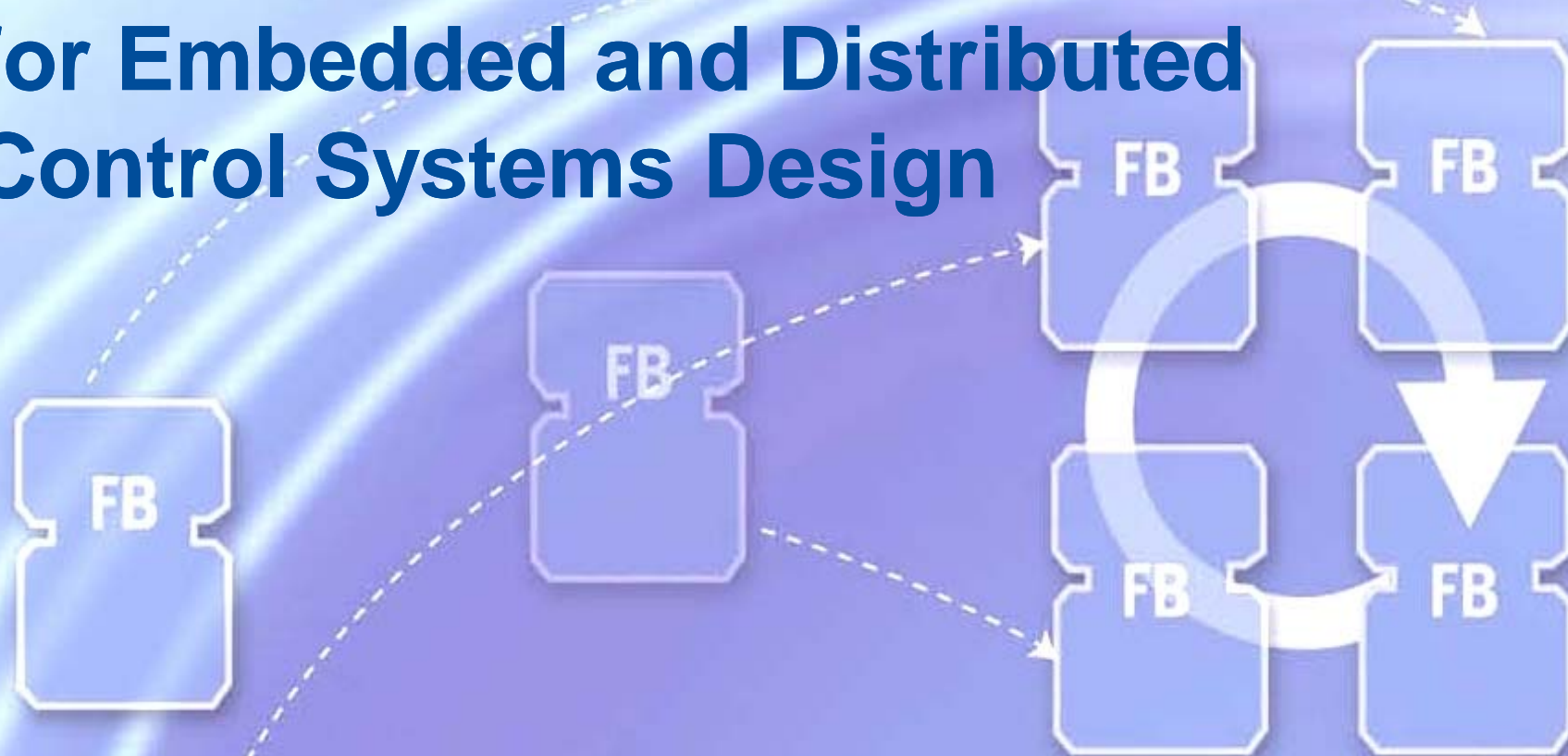


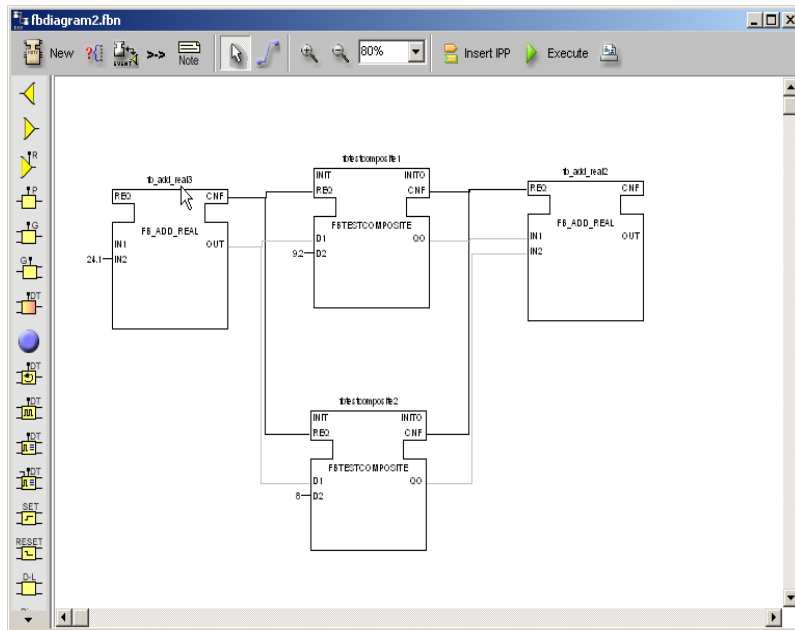
IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design



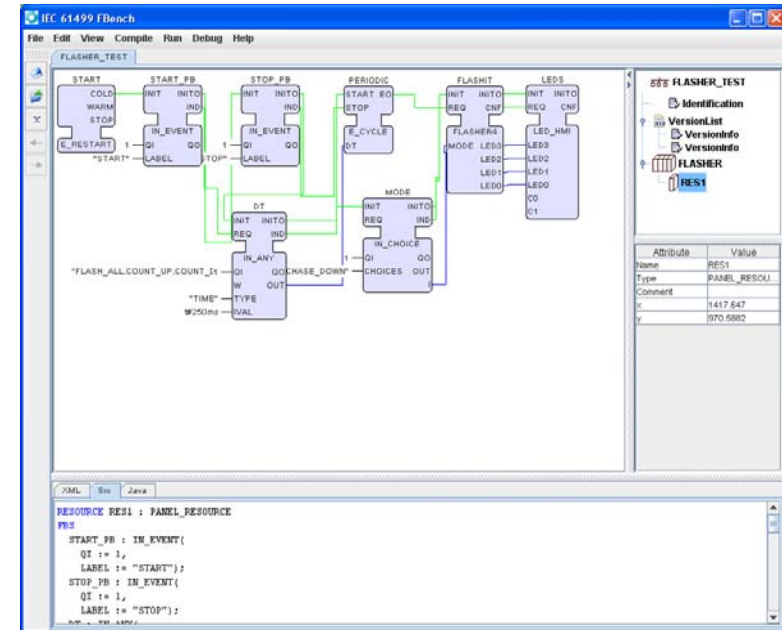
Lecture 6: Function Block Development Kit

Valeriy Vyatkin © 2007

Available Function Block Tools

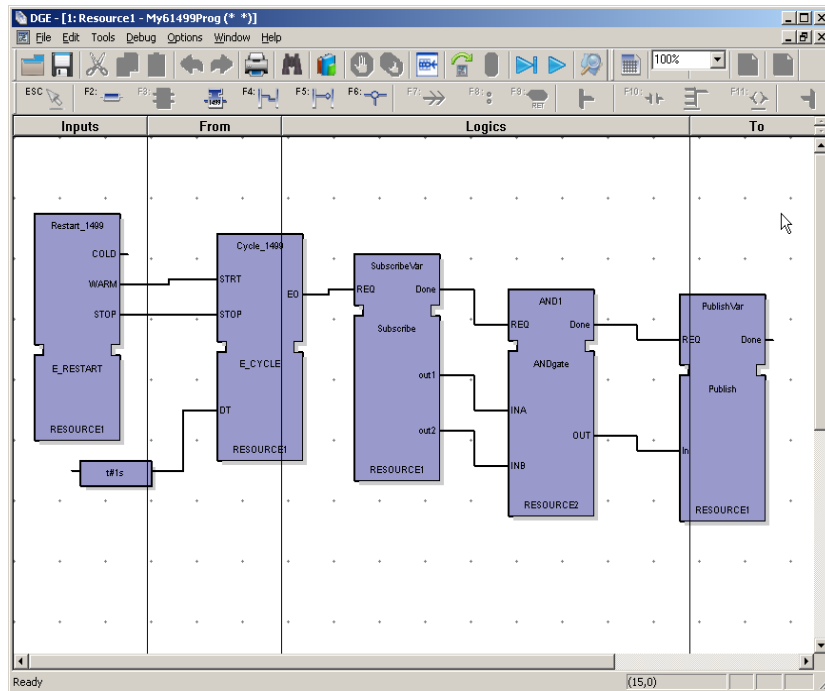


CORFU –University of Patras, Greece
Provides a link between UML and IEC
61499 Function blocks

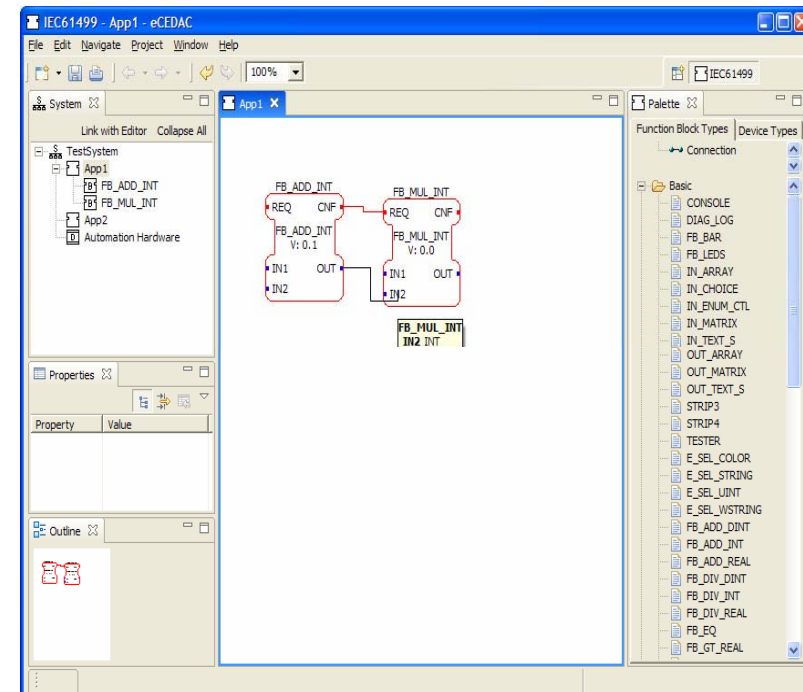


FBench – Open Source development
of the University of Auckland, New
Zealand, based on OOONEIDA
Workbench

Available Function Block Tools

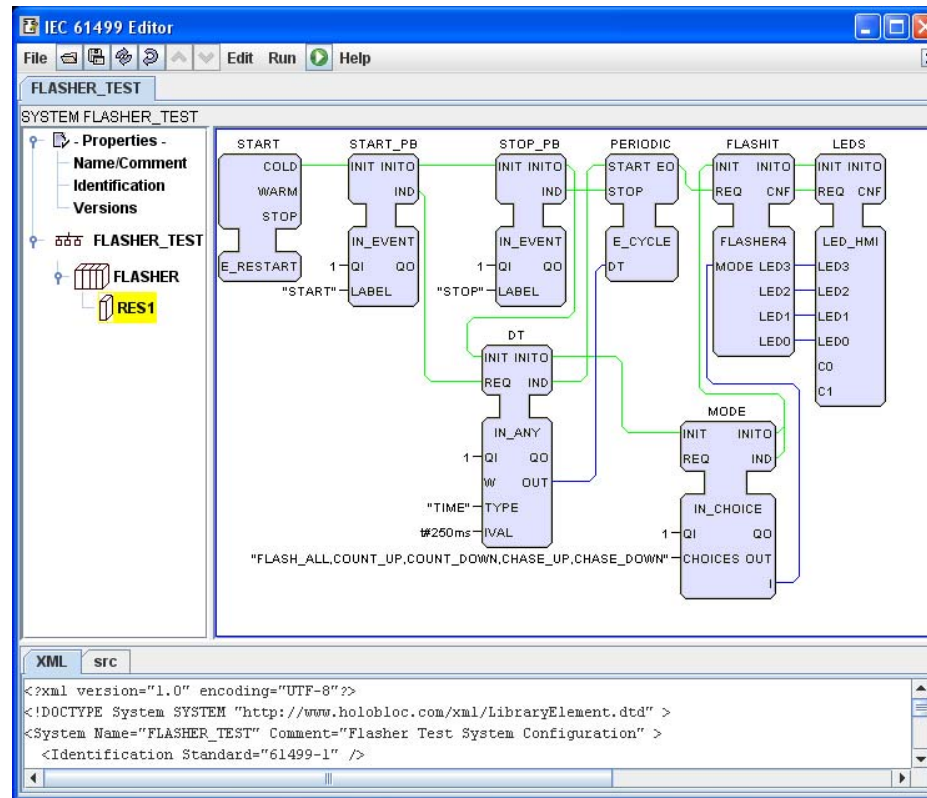


ISaGRAF – first commercial tool,
ICS Triplex, Canada



ODECE –tool implemented as an
Eclipse plug-in, developed at the
Technical University of Vienna,
Austria

Function Block Development Kit (FBDK)



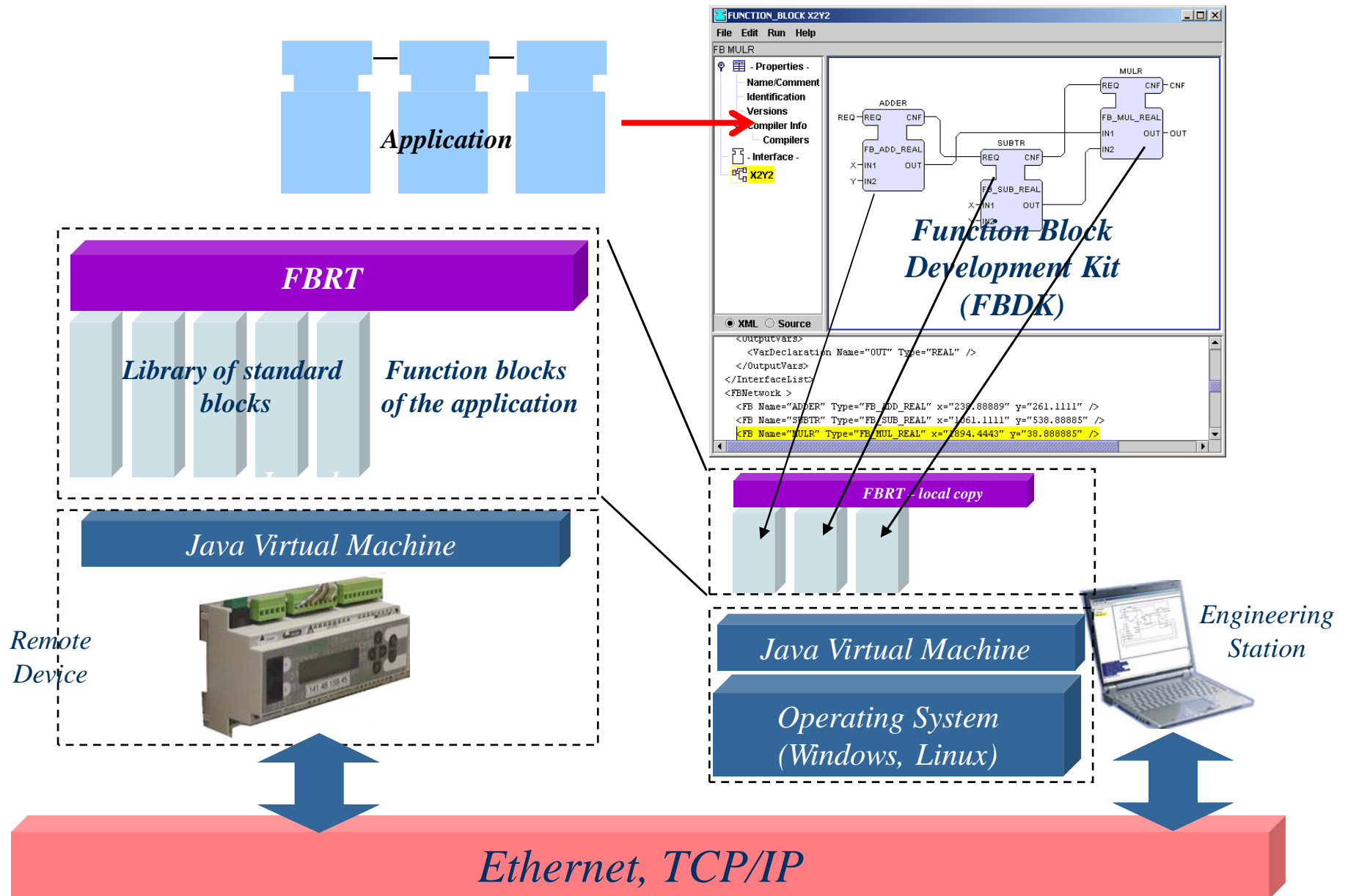
Most popular research tool

Developed by: Rockwell Automation/ Holobloc Inc., USA

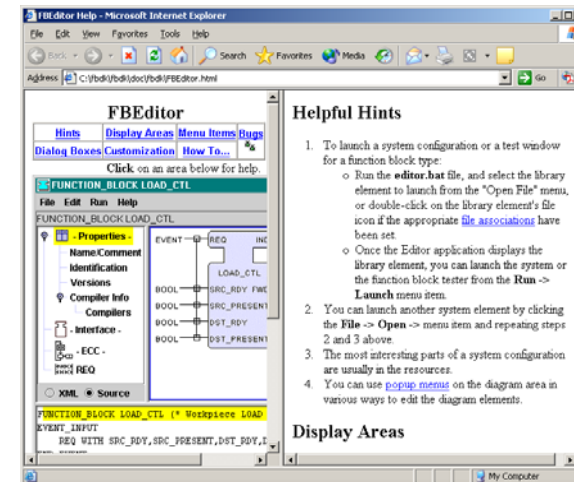
Free downloadable from www.holobloc.com

*Dr. James Christensen –
FBDK Creator and leader
of IEC 61499
Standardisation Working
Group*

FBDK: Java-based implementation of IEC61499

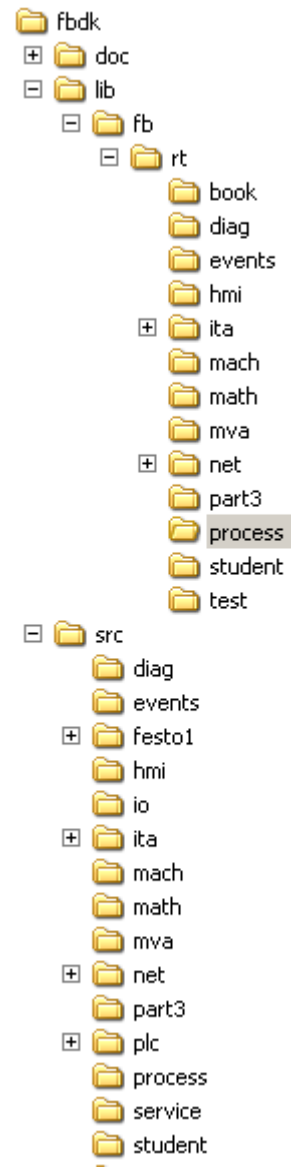


Data Structure



- Documentation in HTML
- Library of JAVA sources and compiled CLASS files
- Library of sources:
 - Function block types (*.fbt)
 - Adapter interface types (*.adp)
 - Resource types (*.res)
 - Device types (*.dev)
 - System configurations (*.sys)
 - ...

Folders



- **Nested according to the Java package hierarchy**
- **lib: fb.rt. + package name**
- **src: package name**

Create a basic FB



- 4. Save *.java
- 5. Compile to *.class

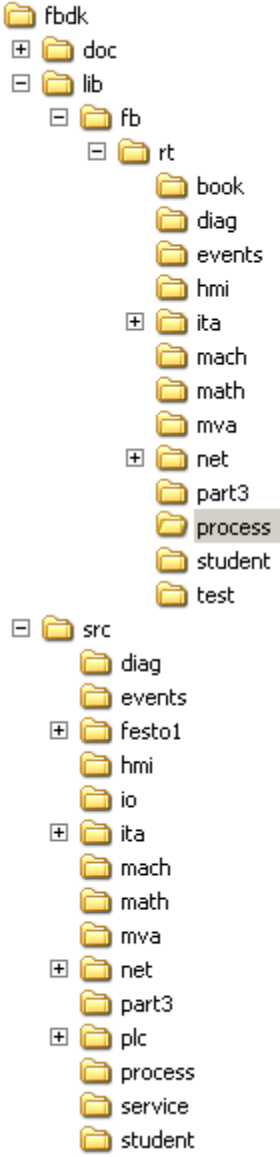
3. Save source

The screenshot shows the 'FUNCTION_BLOCK Basic' software interface. The title bar reads 'FUNCTION_BLOCK Basic'. The menu bar includes 'File', 'Edit', 'Run', and 'Help'. The main window is divided into three sections. The top section displays a diagram of a 'Basic' function block with inputs and outputs: 'EVENT', 'REQ', 'CNF', 'INIT', 'INITO', 'EVENT', 'EVENT', 'EVENT', 'BOOL', 'QI', 'QO', and 'BOOL'. The middle section is a 'Properties' panel with tabs for 'Name/Comment', 'Identification', 'Versions', 'Compiler Info', 'Compilers', 'Interface', and 'ECC'. The bottom section shows XML code for the function block definition.

1. Open template
2. Modify it according to your needs

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE FBType SYSTEM "http://www.holobloc.com/xml/LibraryElement.dtd" >
<FBType Name="Basic" Comment="Basic Function Block Type" >
  <Identification Standard="61499-2" />
  <VersionInfo Organization="Rockwell Automation" Version="0.2" Author="JHC" Date=
  <VersionInfo Organization="Rockwell Automation" Version="0.1" Author="JHC" Date=
  <VersionInfo Organization="Rockwell Automation" Version="0.0" Author="JHC" Date=
  <CompilerInfo header="package fb.rt.template;" >
```


Run Function Block



FUNCTION_BLOCK X2Y2_ST

File Edit Run Help

FUNCTION Launch ST

Log Trace Wait...

Compiler Info Compilers

Interface

ECC

X:=y+2; REQ

XML Source

EVENT REQ CNF EVENT

REAL X OUT REAL

REAL Y

X2Y2_ST

X2Y2_TEST

Restart

REQ CNF

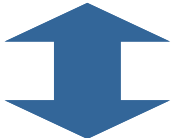
4 7.0

3

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE FBType SYSTEM "http://www.holobloc.com/xml/LibraryElement.dtd">
<FBType Name="X2Y2_ST" Comment="Compute X^2-Y^2 in ST" >
  <Identification Standard="61499-1" />
  <VersionInfo Organization="Rockwell Automation" Version="0.0" Author="JHC" Date="2007-07-07" />
  <CompilerInfo header="package fb.rt.student; " >
    <Compiler Language="Java" Vendor="IBM" Product="Jikes" Version="1.0.6" />
  </CompilerInfo>
  <InterfaceList>
  <EventInputs>

```



Built-in FBRT

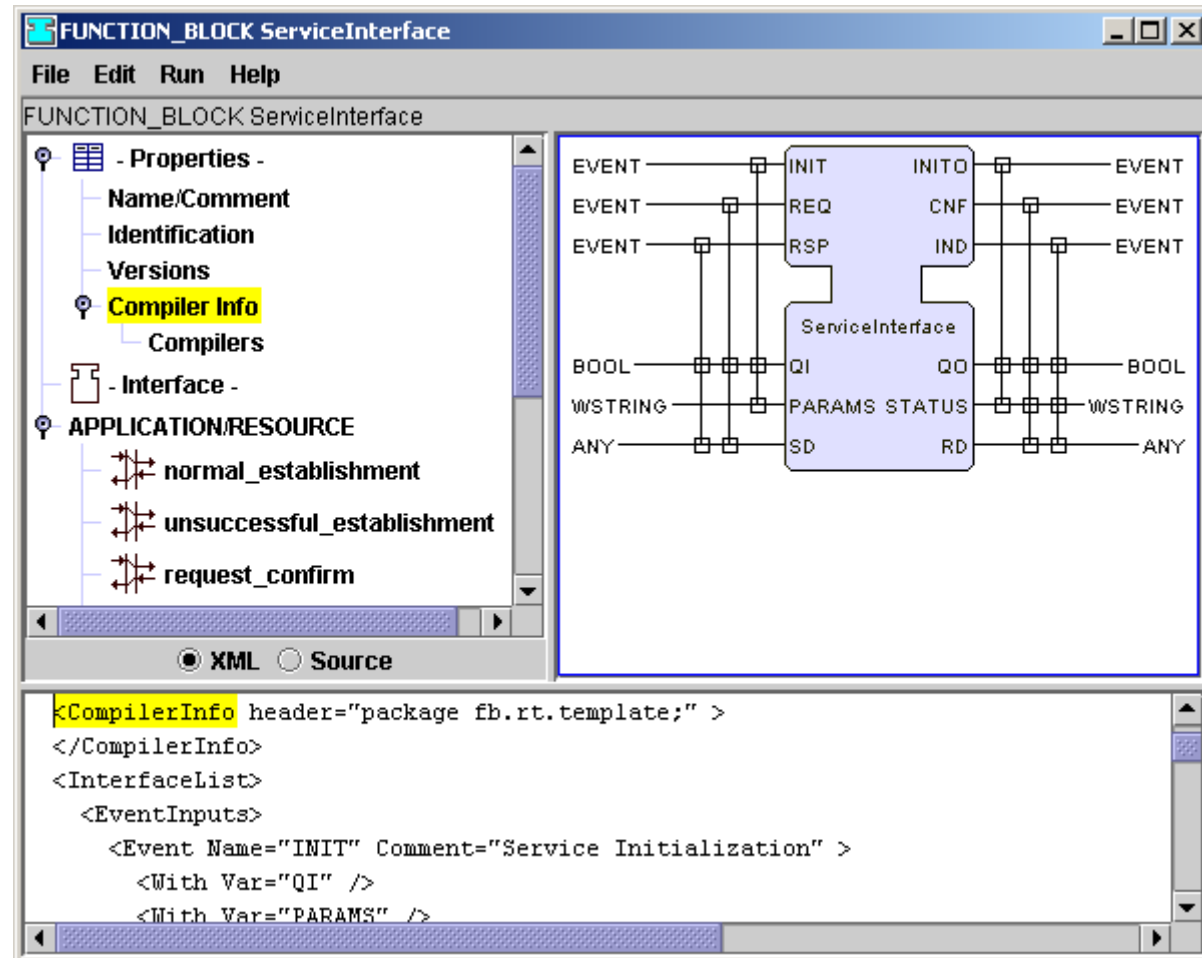
Create a Composite FB (or Application)

The screenshot displays a software development environment with a file explorer on the left and a main workspace. The file explorer shows a directory structure with folders like 'fbdk', 'doc', 'lib', 'fb', 'rt', 'book', 'diag', 'events', 'hmi', 'ita', 'mach', 'math', 'mva', 'net', 'part3', 'process', 'student', 'test', 'src', 'diag', 'events', 'fest01', 'hmi', 'io', 'ita', 'mach', 'math', 'mva', 'net', 'part3', 'plc', 'process', 'service', and 'student'. The main workspace is titled 'FUNCTION_BLOCK strg_fh' and contains a diagram of a composite function block. The diagram shows several sub-blocks: 'Taster_F', 'Taster_M', 'Prioritätssteuerung', 'taster_cmd', and 'E_DELAY'. The 'Taster_F' block has inputs 'INDI' and 'REQ' and outputs 'req0'. The 'Taster_M' block has inputs 'INDI' and 'REQ' and outputs 'req1'. The 'Prioritätssteuerung' block has inputs 'REQ_0', 'REQ_1', and 'REQ2' and outputs 'cnf0', 'cnf1', and 'req0'. The 'taster_cmd' block has inputs 't0_oeffnen' and 't0_schliessen' and outputs 'kom0'. The 'E_DELAY' block has inputs 'START EQ' and 'STOP' and outputs 'E_DELAY'. The diagram is annotated with red text: '1. Open template' points to the 'Taster_F' block, '2. Populate the FB network with FB instances according to your needs' points to the connections between blocks, '4. Save *.java' points to the 'ita' folder in the file explorer, and '5. Compile to *.class' points to the 'src' folder in the file explorer. Below the diagram, the XML source code is displayed:

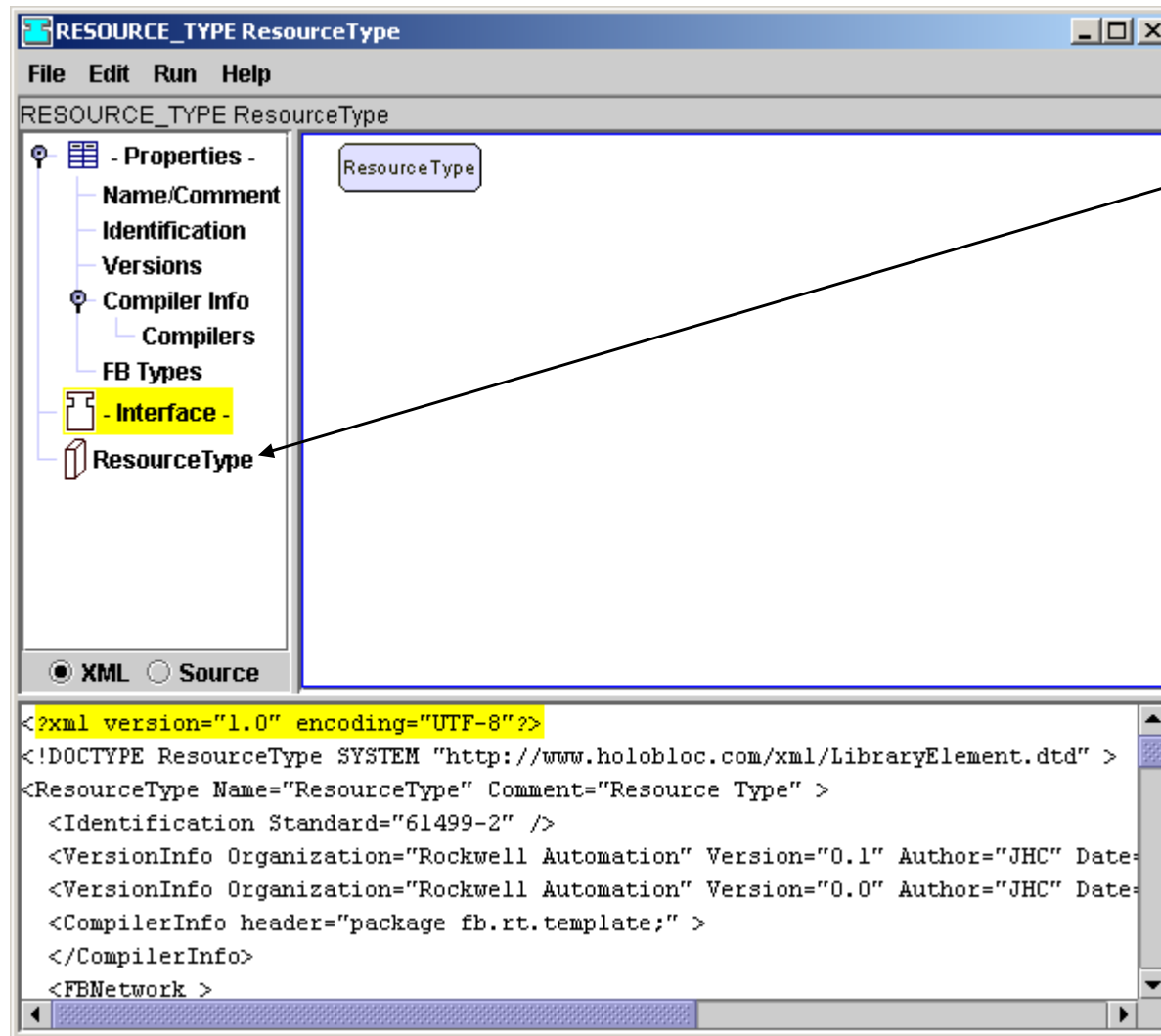
```
<VarDeclaration Name="tl_schliessen" Type="BOOL" />
<VarDeclaration Name="f_geoeffnet" Type="BOOL" />
<VarDeclaration Name="f_geschlossen" Type="BOOL" />
</InputVars>
```

Create a SIFB

1. Start with the template
2. Modify the interface
3. Modify the list of implemented services
4. Save as *.fbt
5. Save as *.JAVA
6. Modify manually JAVA code adding new methods implementing the services
7. Compile *.JAVA into *.Class



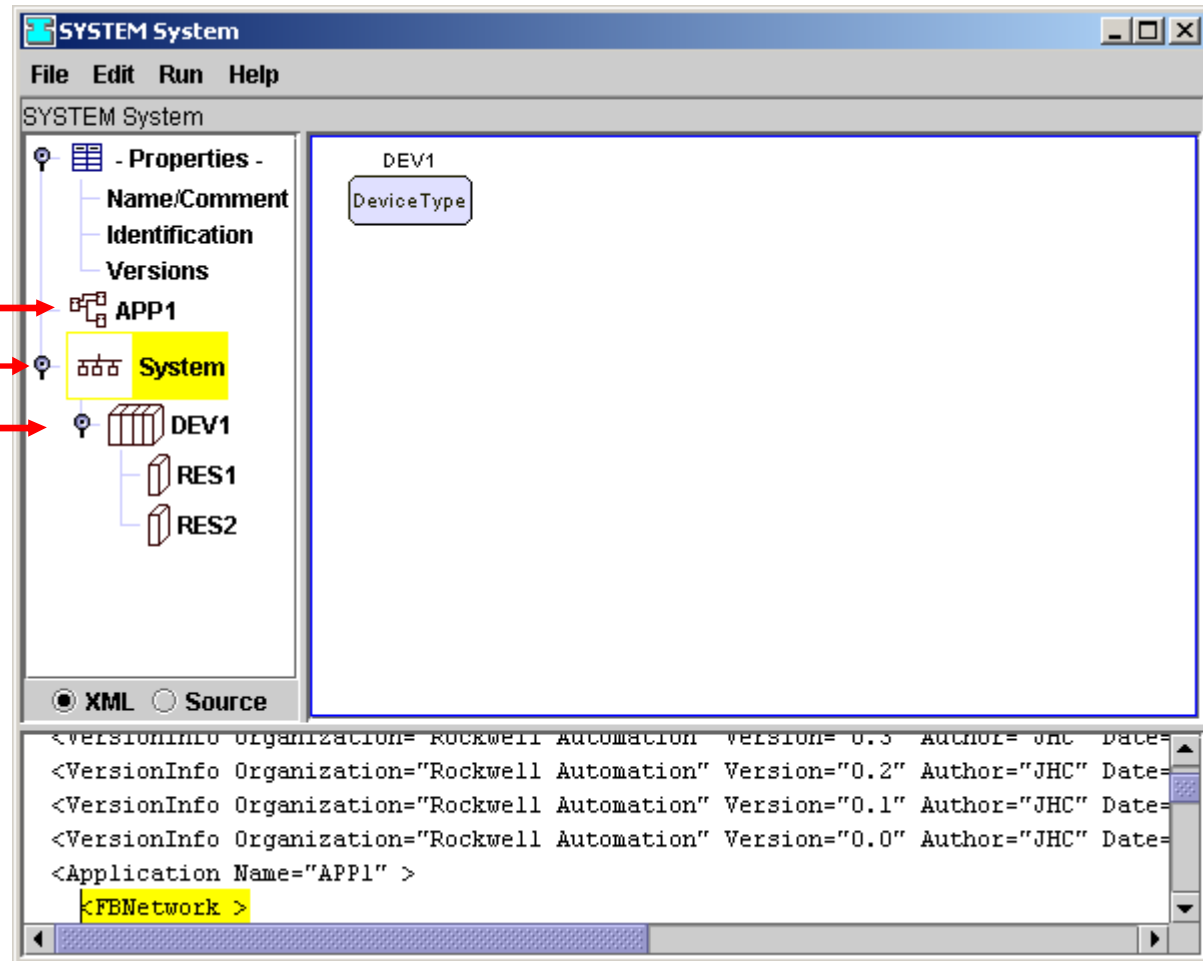
Create a resource type



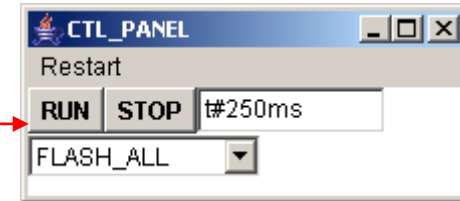
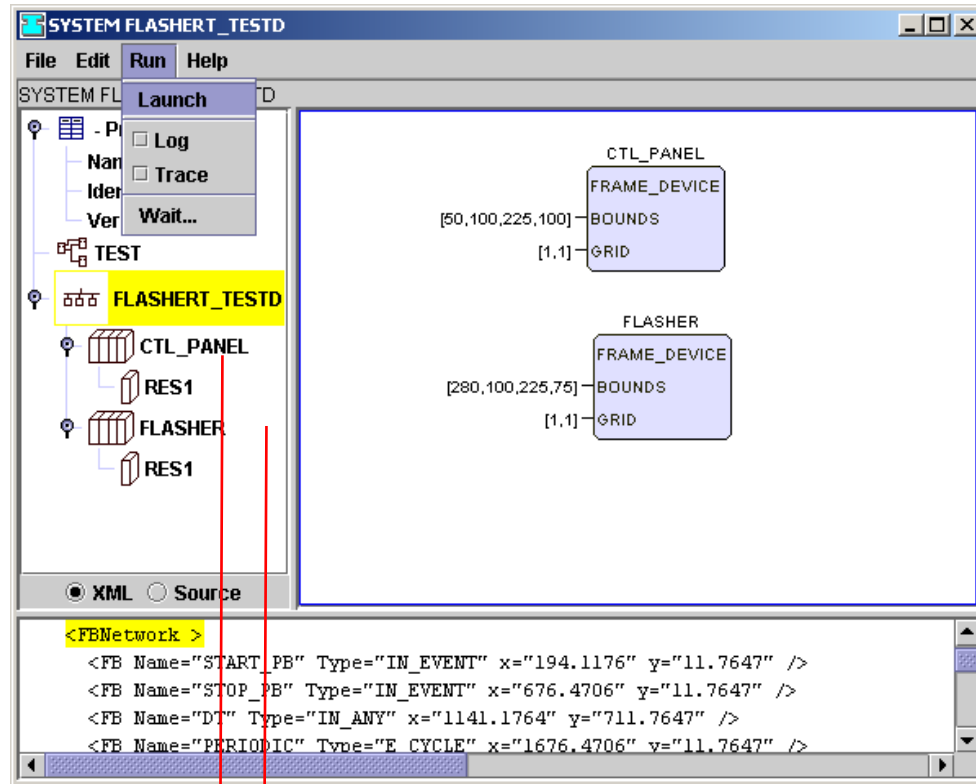
1. Start with resource template
2. Populate with function block instances needed in this resource by default (other function block instances may be added to the resource instance)
3. Save as *.res
4. Save as *.java
5. Compile *.java into *.class

Create a System Configuration

1. Start with the system template
2. Add/create the application
3. Modify/Create the structure of devices by instances of device types
4. Map function blocks from the application to devices
5. Parameterize device instances
6. Save as *.sys
7. RUN!



Run System Configuration



Create device instance

Built-in FBRT

Run a Distributed System Configuration

The image displays a software interface for configuring a distributed system. The main window, titled "SYSTEM FLASHERT_TESTR", shows a hierarchical tree on the left with "FLASHERT_TESTR" selected. The central area contains a diagram of the system components: a "CTL_PANEL" block with sub-components "FRAME_DEVICE", "BOUNDS" (with values [50,100,325,100]), and "GRID" (with value [1,1]); and a "FLASHER" block with sub-components "RMT_DEV" and "MGR_ID". The "MGR_ID" component is associated with the value "localhost:61501".

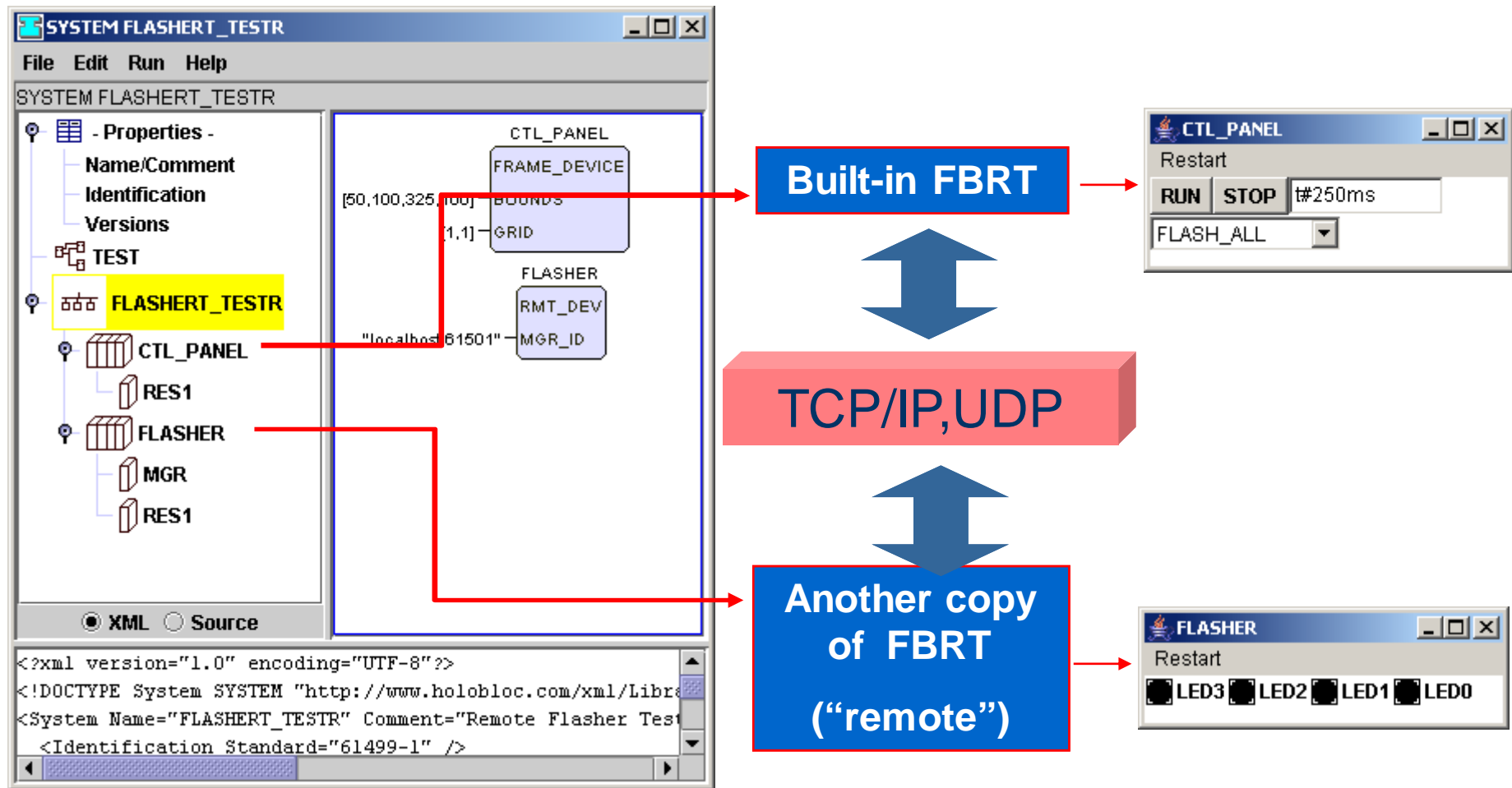
To the right, a file explorer window shows the directory structure of the "fbdk" project, including folders like "doc", "lib", "src", "diag", "events", "festo1", "hmi", "io", "ita", "ivo_matz", "mach", "math", "mva", "net", "parts", "plc", and "process".

At the bottom, a Notepad window titled "remote.bat" contains the following command:

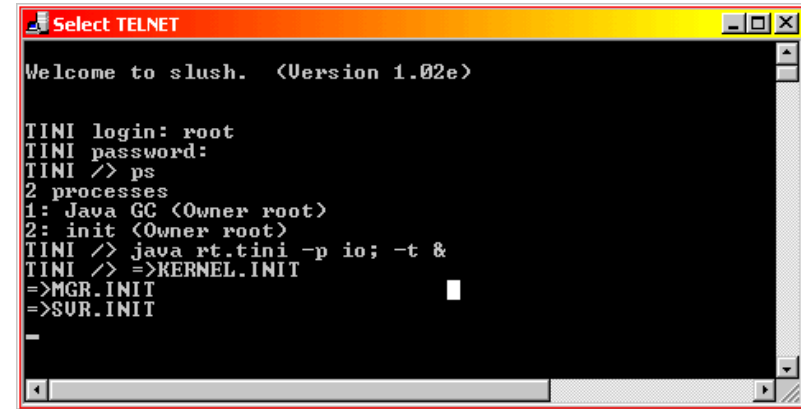
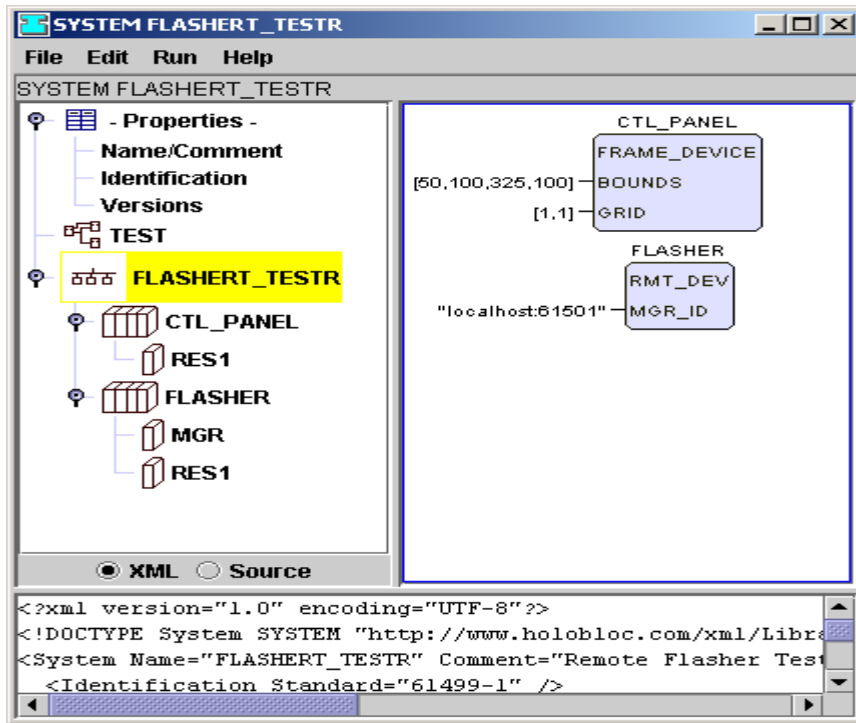
```
java -noverify -classpath lib;fbdt.jar; RMT_DEV -name RDEVICE -p math;ita; -s 61501
```

Red arrows point from the text "parameterized according to the parameters given in system configuration" to the "localhost:61501" value in the diagram and the "-s 61501" parameter in the command prompt.

Run a Distributed System Configuration (2)



Run a remote device

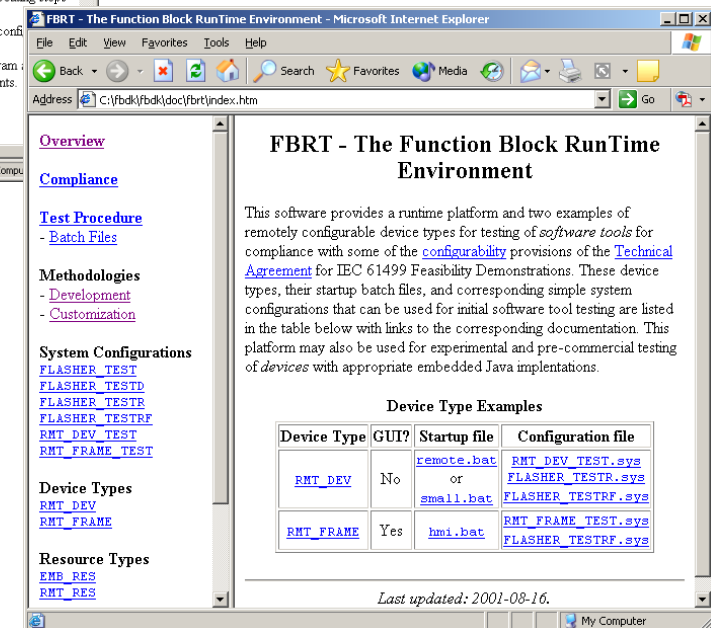
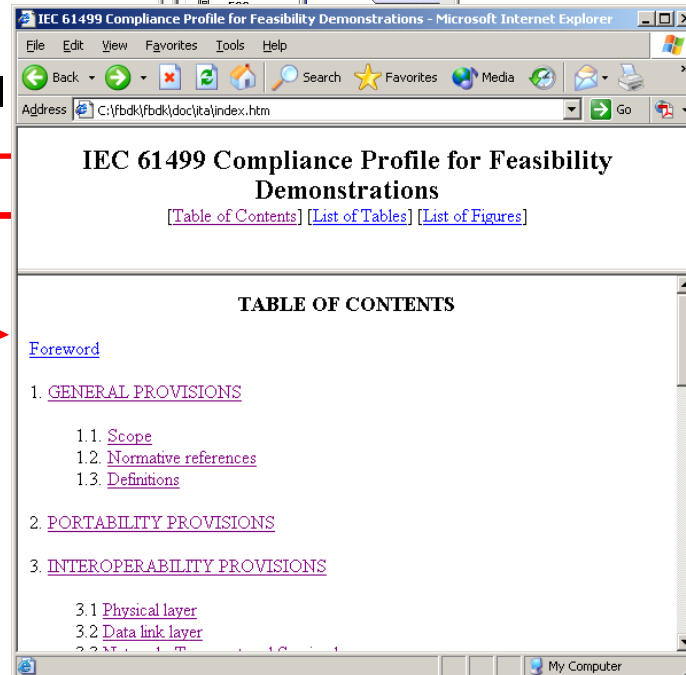
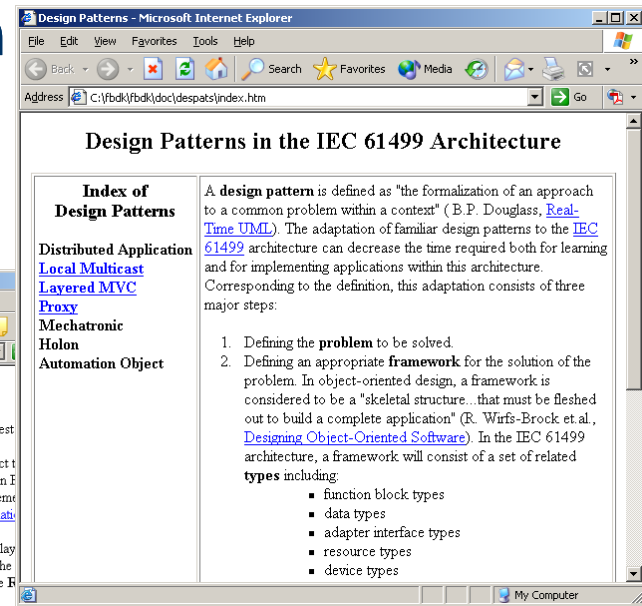
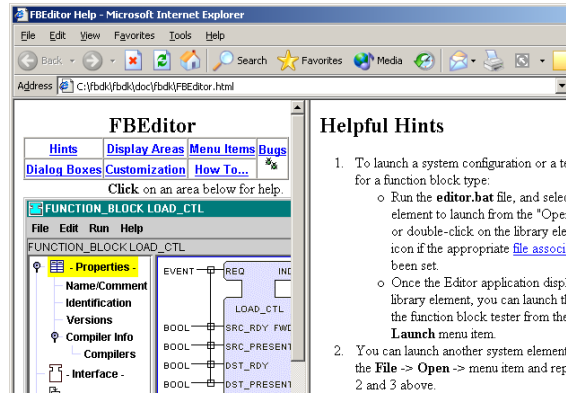
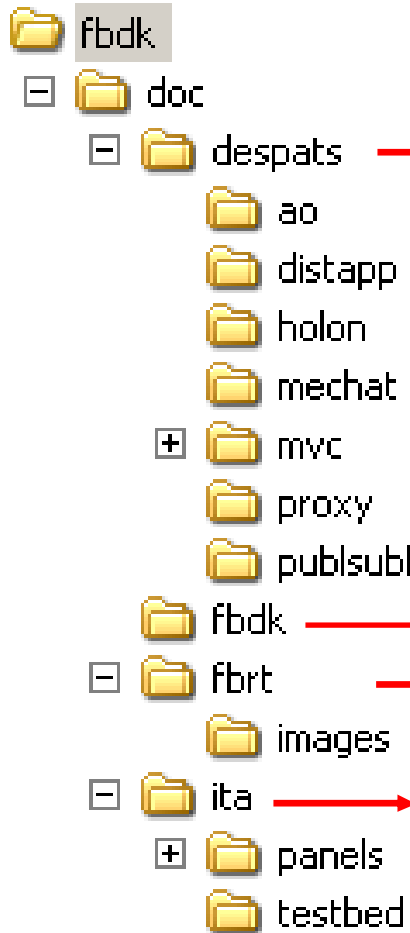


Remote FBRT

Built-in FBRT



Documentation



What is missing in FBDK?

- **Build the system configuration (recompile necessary function blocks)**
- **Dynamic trace of data/algorithms**
- **Dynamic upload of classes into running remote devices (requires also support from the device side)**