Lecture 3
ASU VIPLE
Visual IoT/Robotics Programming Language Environment

Dr. Yinong Chen

IRA A. FULTON SCHOOLS OF engineering

school of computing, informatics, & decision systems engineering

Lecture Outline

1. Introduction to VIPLE
2. General-Purpose Programming
3. Service-Oriented Programming
4. Parallel Programming
5. Event-Driven Programming

Application of Visual Programming Languages

- Most workflow languages today are visualized;
- Simplified workflow languages are used in education:
  - MIT: Scratch - Visual Game Programming
  - University of Virginia and Carnegie Mellon: Alice Visual Game Programming
  - MIT App Inventor: Phone App Visual Programming
  - Lego NXT & EV3 – Visual Robotics Application Development
  - Microsoft Robotics Developer Studio Visual Programming Language (MRDS VPL)
  - ASU **VIPLE**: Visual IoT/Robotics Programming Language Environment

ASU VIPLE Download Site

- Download Link: http://venus.eas.asu.edu/WSRepository/VIPLE/

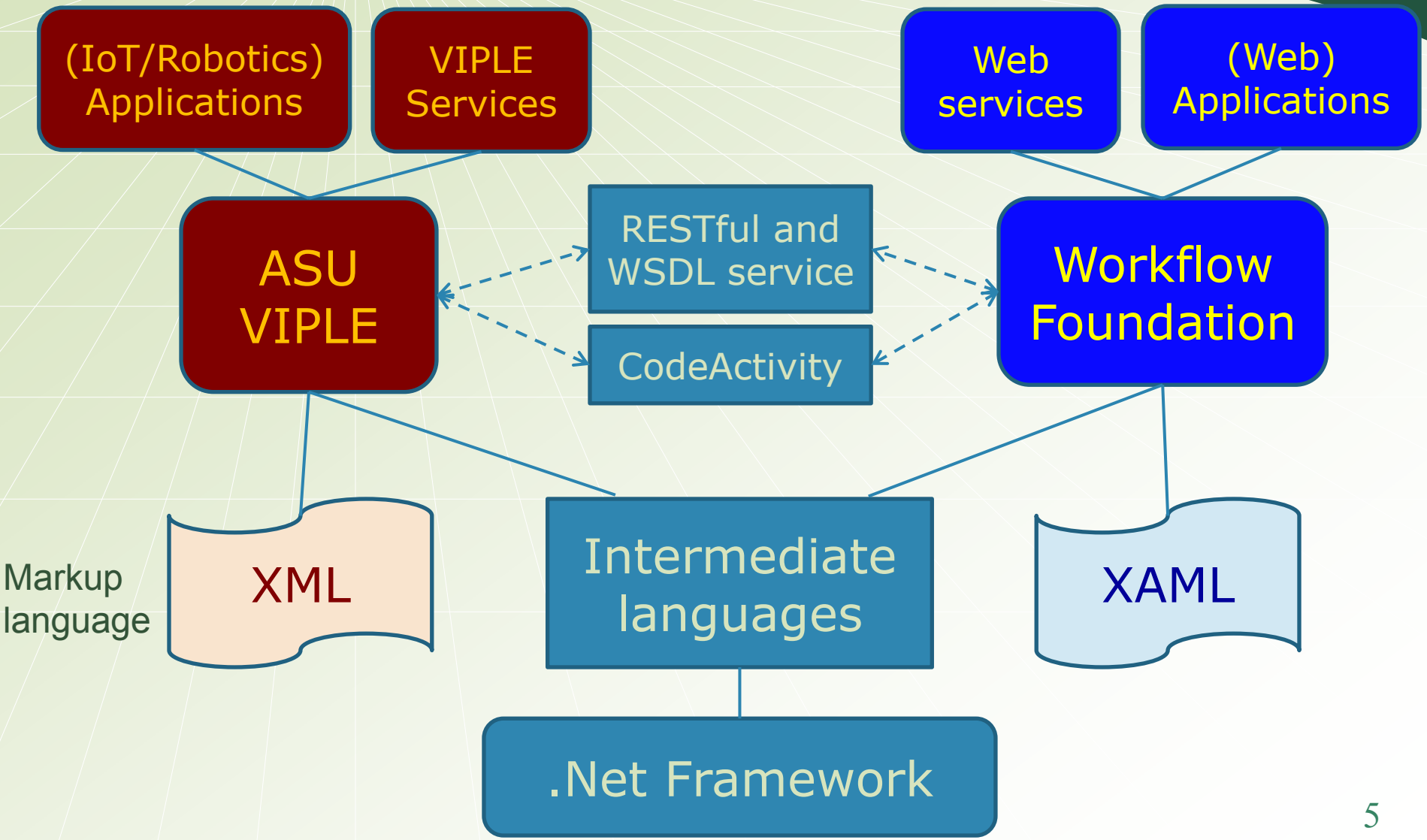
venus.eas.asu.edu/WSRepository/VIPLE/

VIP VISUAL IOT/ROBOTICS PROGRAMMING LANGUAGE ENVIRONMENT

ASU VIPLE Download and Resources

- **ASU VIPLE Documents**
  - ASU VIPLE Introduction: ASU VIPLE Tutorial
  - Textbook: Service-Oriented Computing and Web Software Integration
  - Repository: ASU Repository of Web Services and Web Applications
- **ASU VIPLE and Edison Middleware Downloads**
  - ASU VIPLE Software Download. Unzip the file and start the application from file: VisualPr
  - Intel Edison Board Installer: A link to Intel Website. The site will instruct you to install Lin VIPLE Middleware to run. ASU VIPLE middleware will communicate with ASU VIPLE on you use different sensors and motors for your robot.
  - ASU VIPLE Middleware on Edison. Unzip the file and copy all the folders and files into Ed
  - Basic Sample Programs Written in ASU VIPLE
  - Full Sample Programs Written in ASU VIPLE for Instructors only. Please contact Dr. Chen
- **ASU VIPLE Videos**
  - EV3 Wall-Following, with self-adjustment: Link to Video at ASU and Video file download
  - EV3 Line Follower: Link to Video and Video file download
  - Edison Robot Maze Navigation by measure right and left distances and Video file download

VIPLE vs. Workflow Foundation



VIPLE Programming Paradigms

VIPLE Features

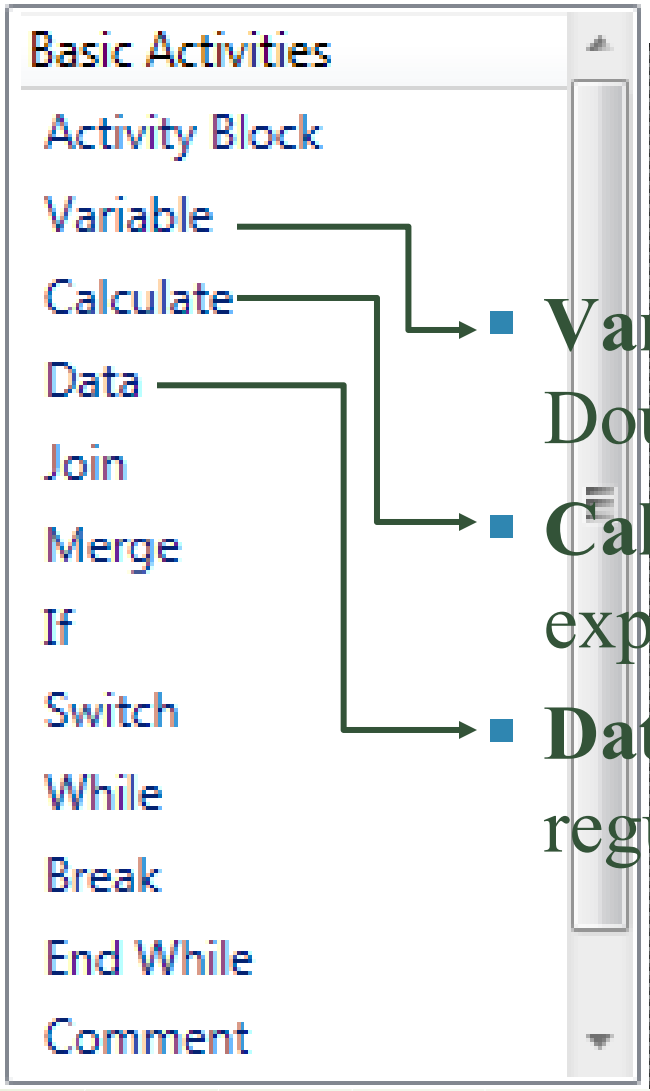
- General-purpose control flow programming
- Service-oriented computing, supporting RESTful and WSDL
- Parallel / multithreading programming, with underlying threads safety
- Event-driven programming, with built-in and custom events
- Workflow and visual programming
- IoT and Robotics programming

Two Main Purposes

- An example of software integration: After taking CSE445 and CSE446, ASU undergraduate students can create such tools.
- A real tool for high school and university students to learning the first programming language

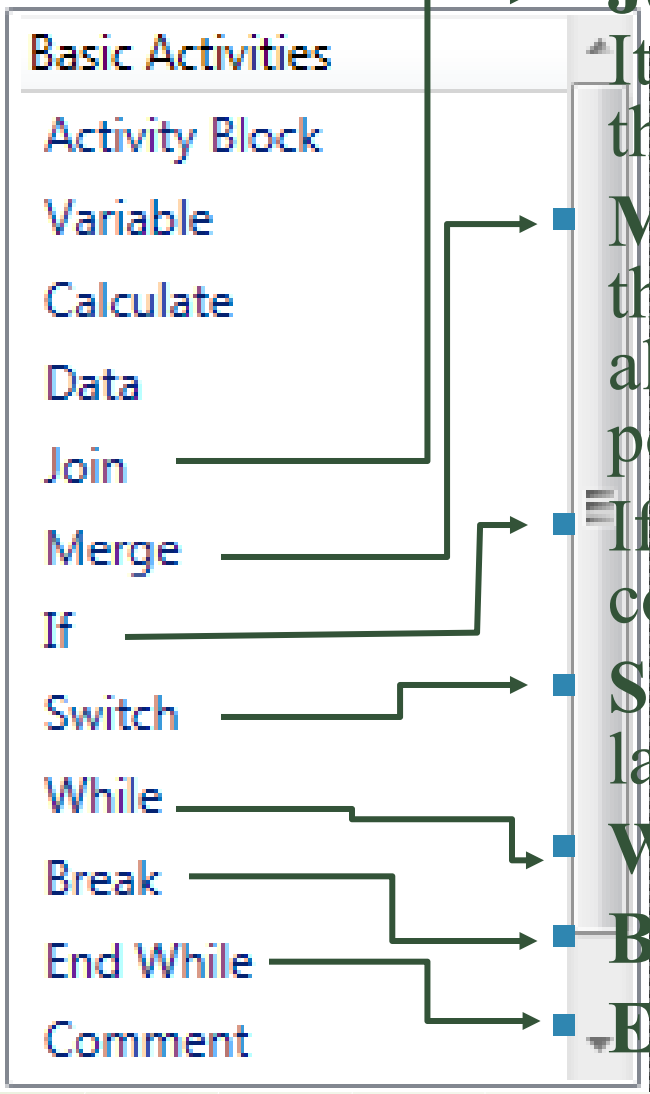
Basic Activities of VIPLE

There are dozen of basic activities, and many composite services in VIPLE Repository



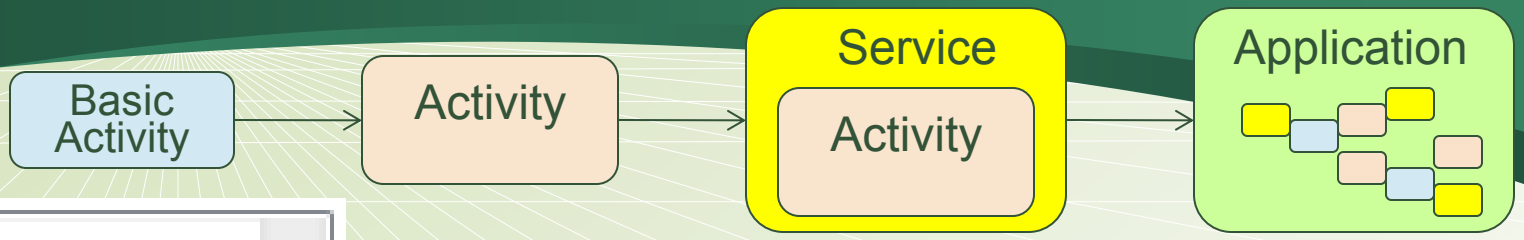
- **Variable**: supports basic types (Int32, Double, String, Boolean, etc.).
- **Calculate**: Calculate the value of any C# expression.
- **Data**: Introducing the constant values in regular programming language

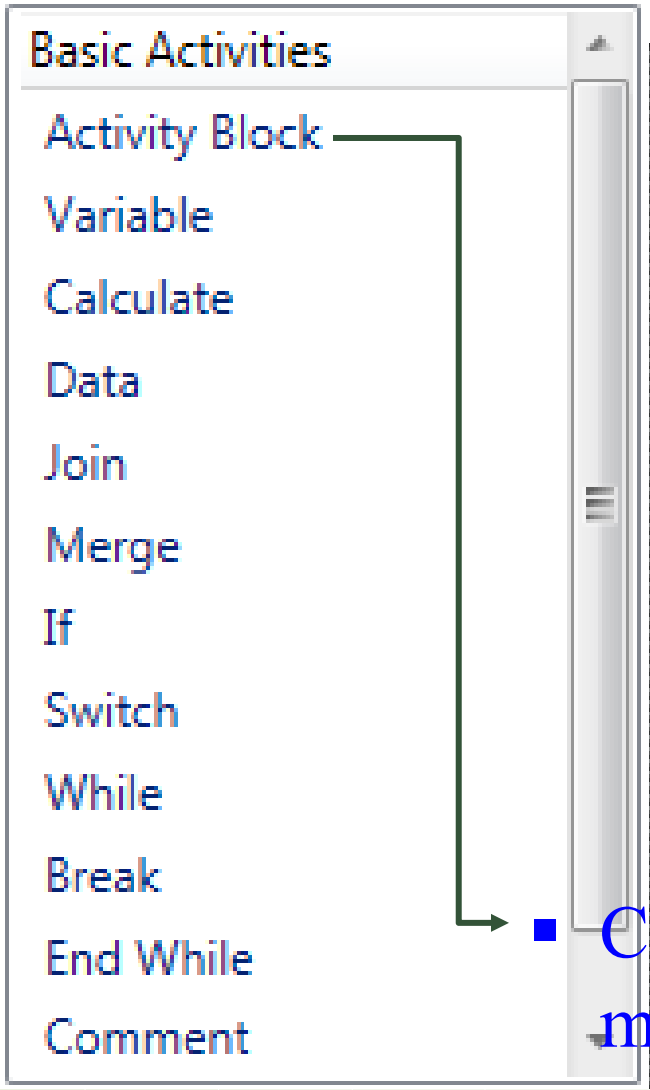
Basic Activities (cont.)



- **Join**: proceeds when all threads arrive; It can be used for waiting parallel data or threads.
- **Merge**: proceeds when one of the data or threads arrives. It can be used for waiting alternative data, e.g., creating the returning point of a loop;
- If: same as regular programming language construct; It allows multiple conditions.
- **Switch**: same as regular programming language construct;
- **While**: start a loop;
- **Break** exits a loop, and
- **End While** returns to While

Basic Activities (cont.)





- Basic Activity and Activity are building blocks of a diagram or flowchart.
- Activity can be wrapped into a service
- Data transfer between
  - activities are through global variables and parameter passing
  - services don't support global

- Construct a composite activity (block or module)

VIPLE Services

General-purpose and event services

Services

- Code Activity
- Custom Event
- Key Press Event
- Key Release Event
- Print Line
- Random
- RESTful Service
- Simple Dialog
- Text to Speech
- Timer

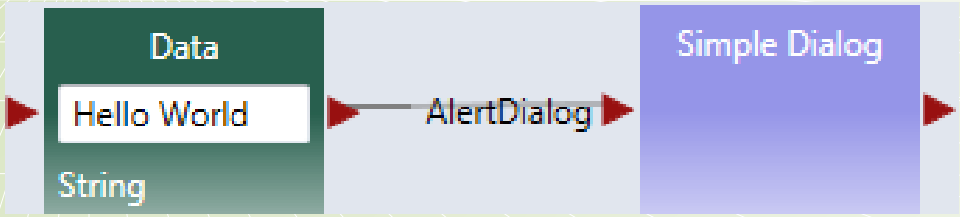
Generic robotic services

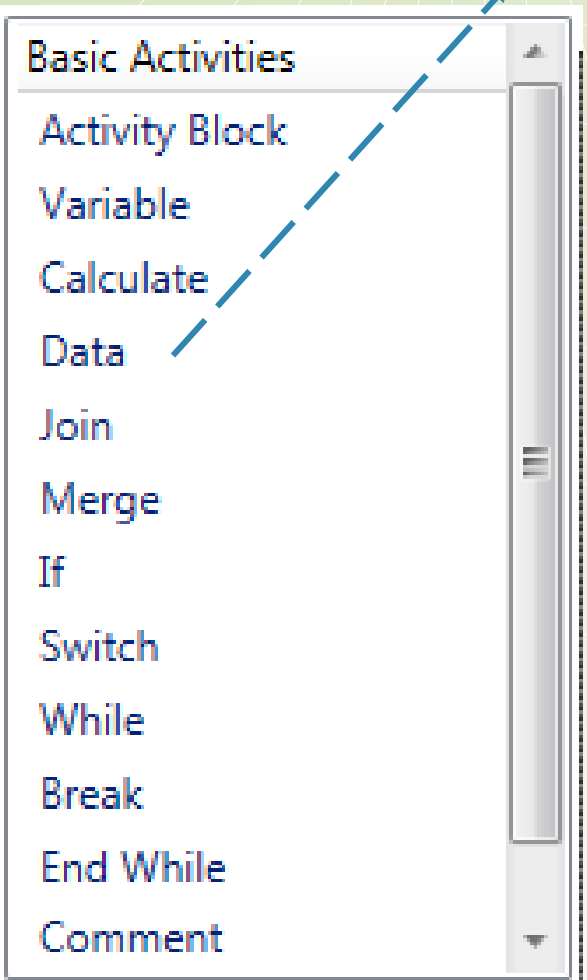
- Robot
- Robot Color Sensor
- Robot Distance Sensor
- Robot Drive
- Robot Holonomic Drive
- Robot Light Sensor
- Robot Motor
- Robot Motor Encoder
- Robot Sound Sensor
- Robot Touch Sensor
- Robot+ Move at Power
- Robot+ Turn by Degrees

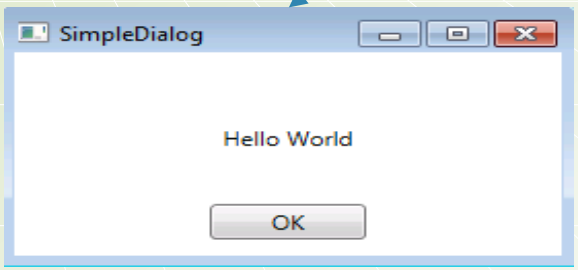
Vendor-specific robotic services

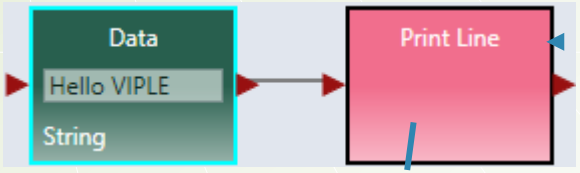
- Lego EV3 Brick
- Lego EV3 Color
- Lego EV3 Drive
- Lego EV3 Drive for Time
- Lego EV3 Gyro
- Lego EV3 Motor
- Lego EV3 Motor by Degrees
- Lego EV3 Motor for Time
- Lego EV3 Touch Pressed
- Lego EV3 Touch Released
- Lego EV3 Ultrasonic

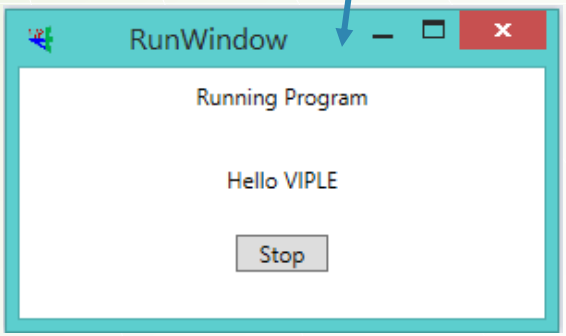
VIPLE Programming: Output



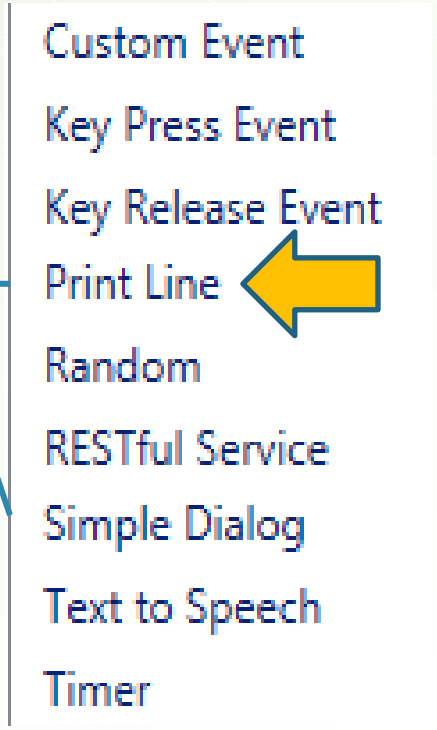




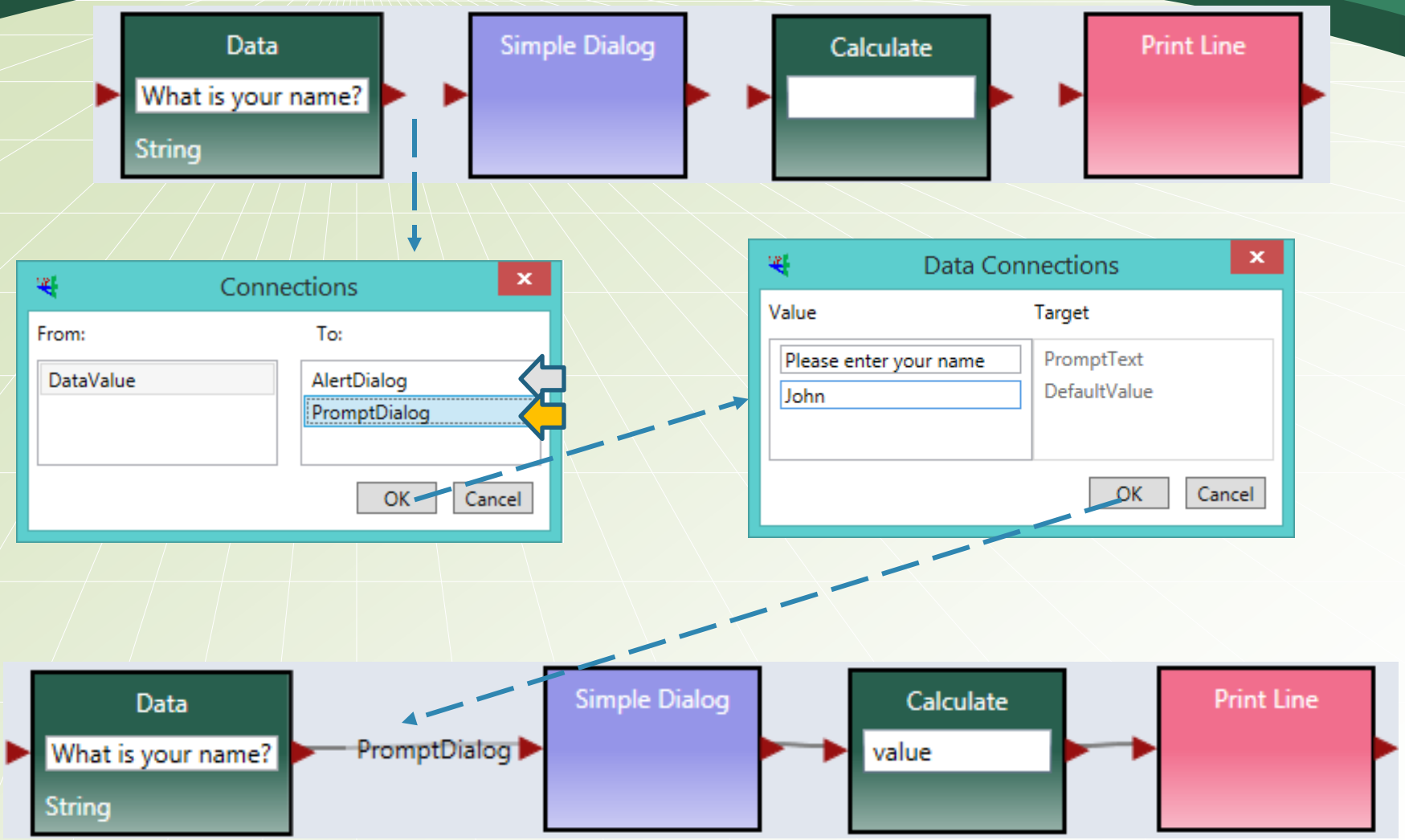




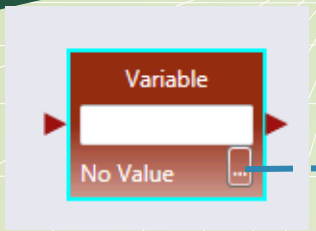
General-purpose and event services

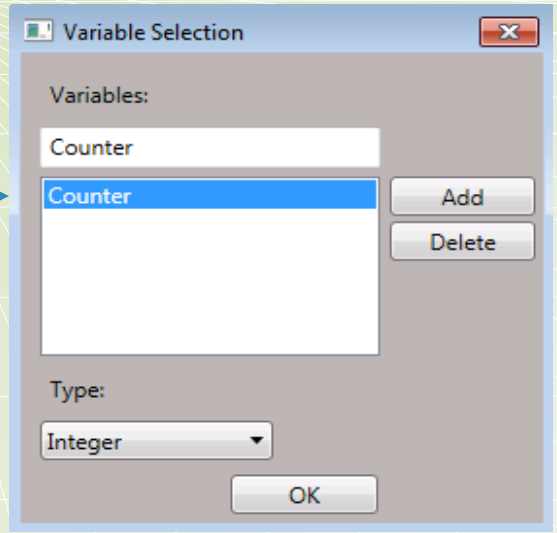


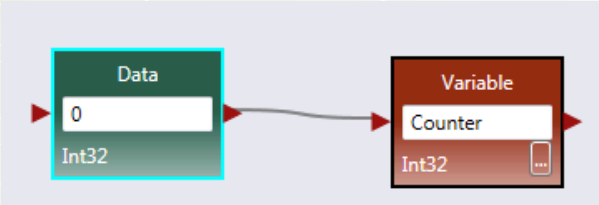
VIPLE Programming: Input

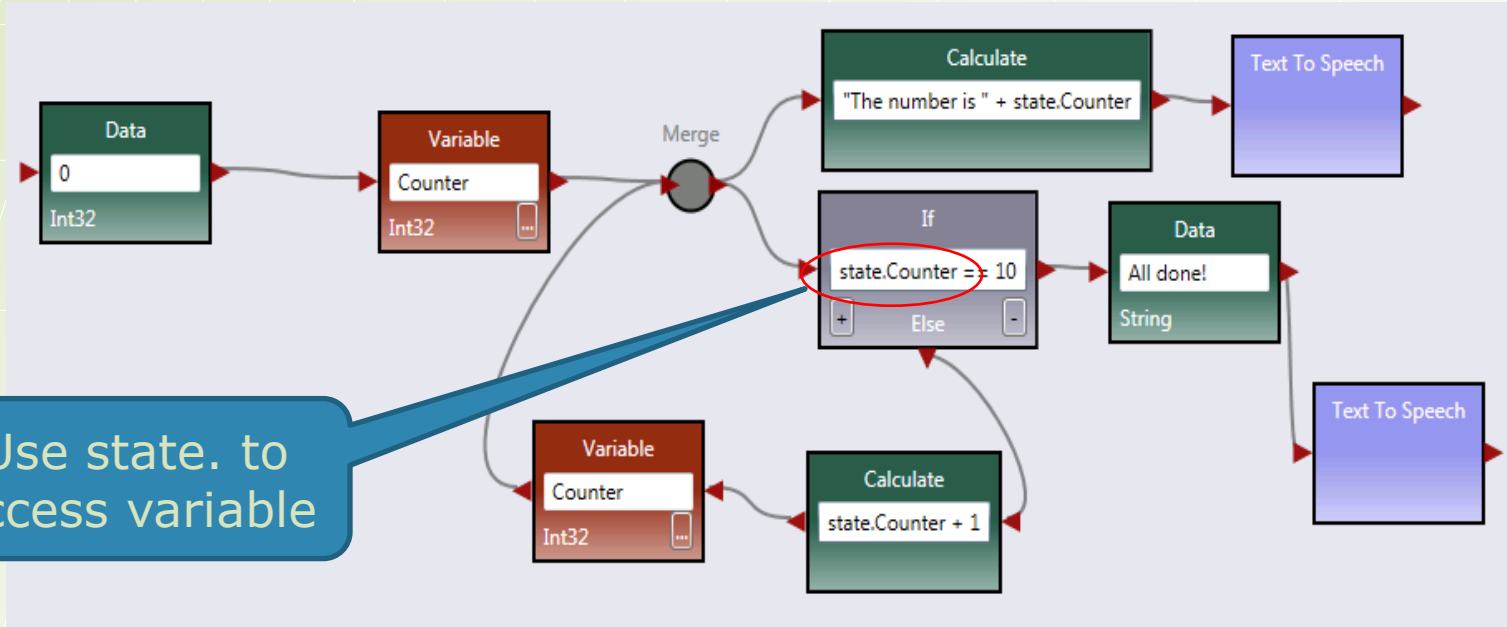


VIPLE Programming: Variable and Loop



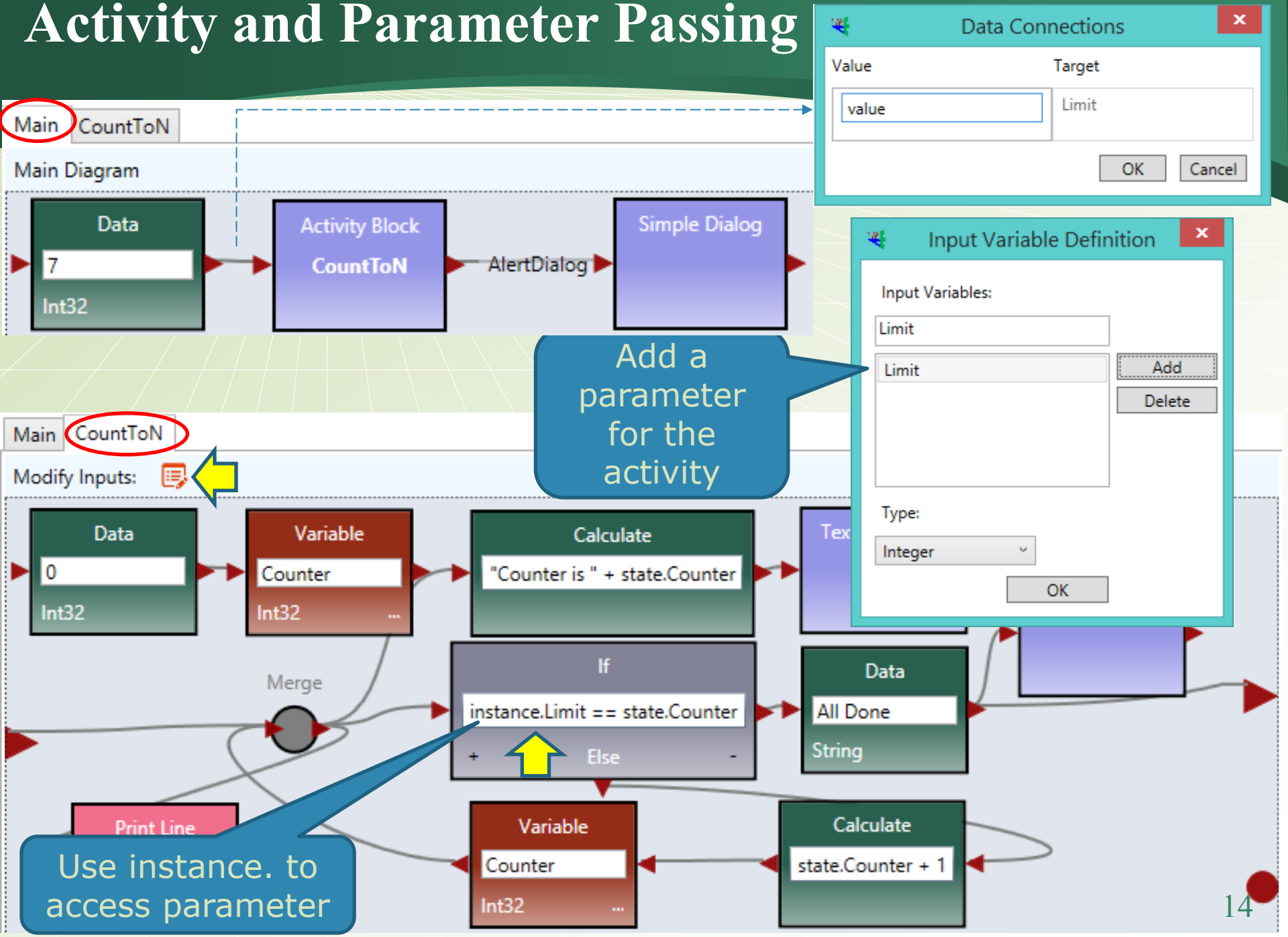




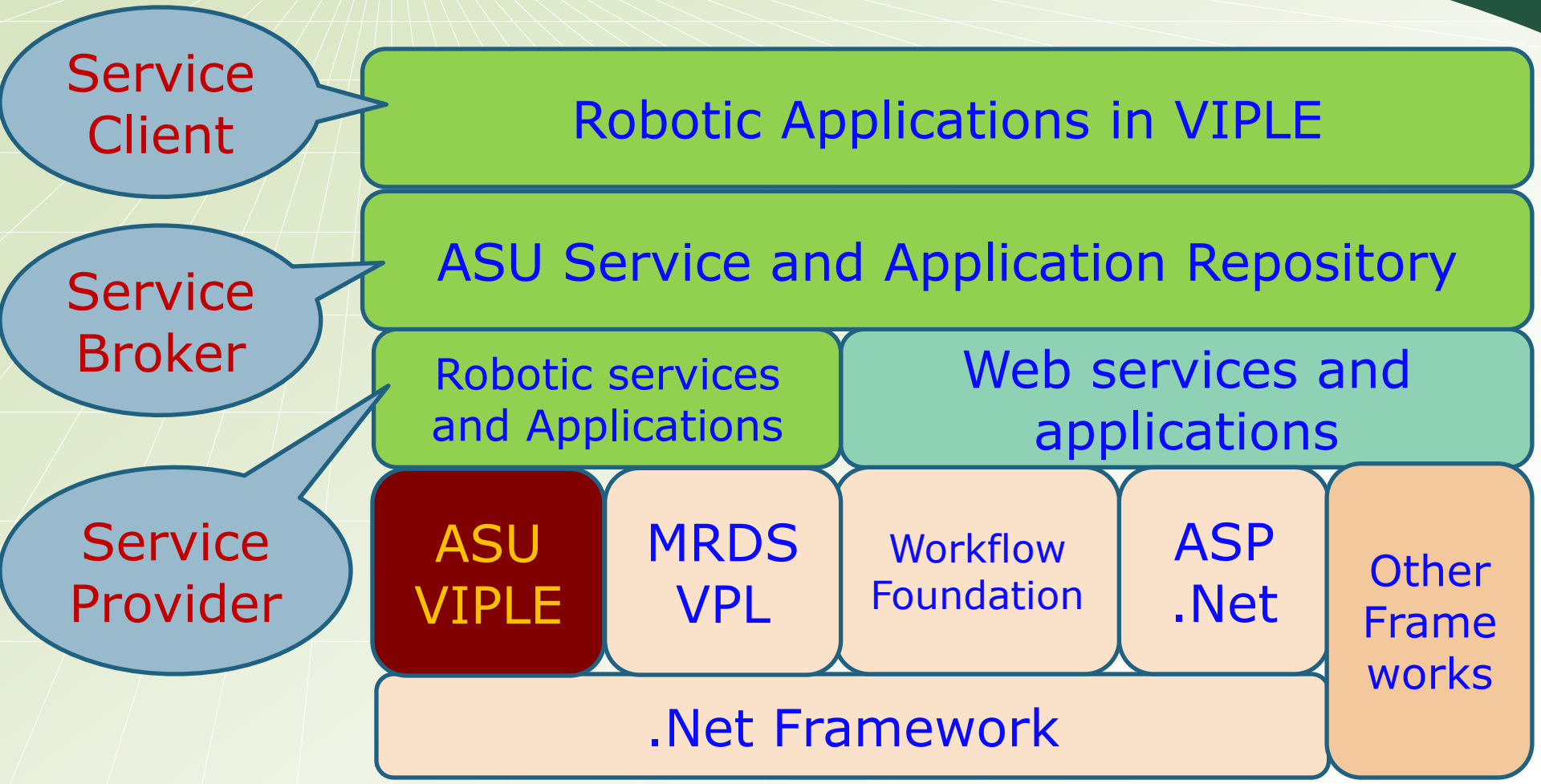


Use state. to access variable

Activity and Parameter Passing

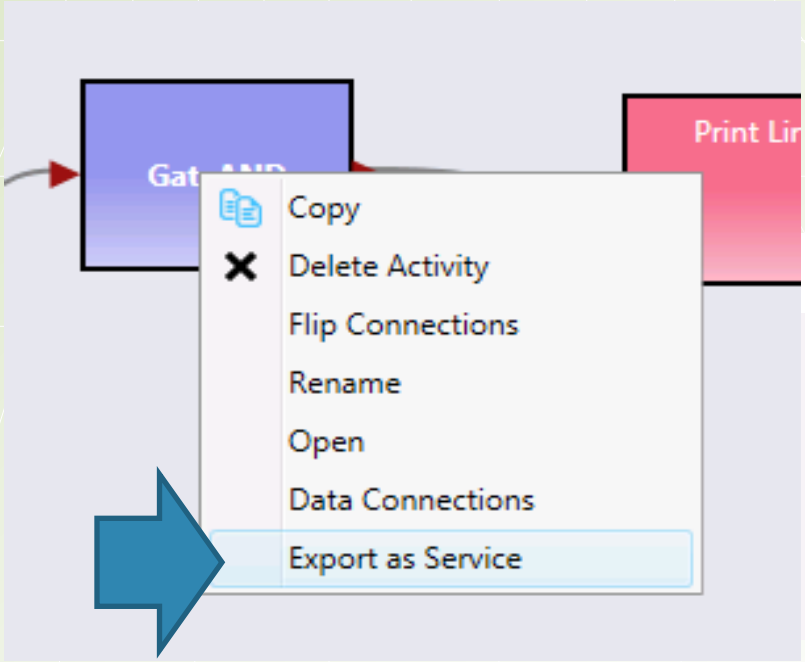


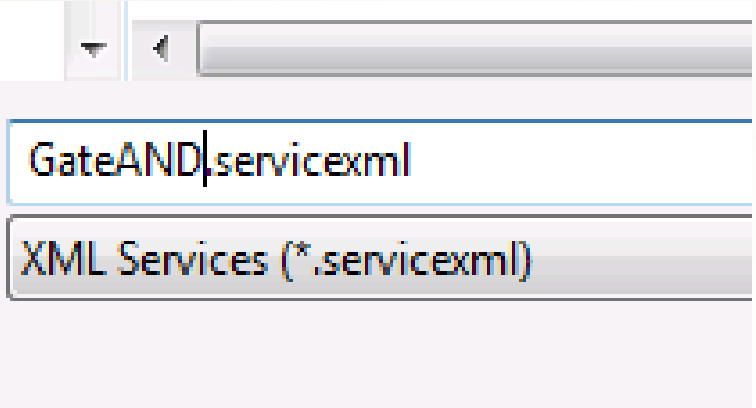
ASU VIPLE is Service Oriented



Converting an Activity into a Service

- An activity is a part of an application
- It cannot be reused in another application
- To convert an activity into a service: Right click:

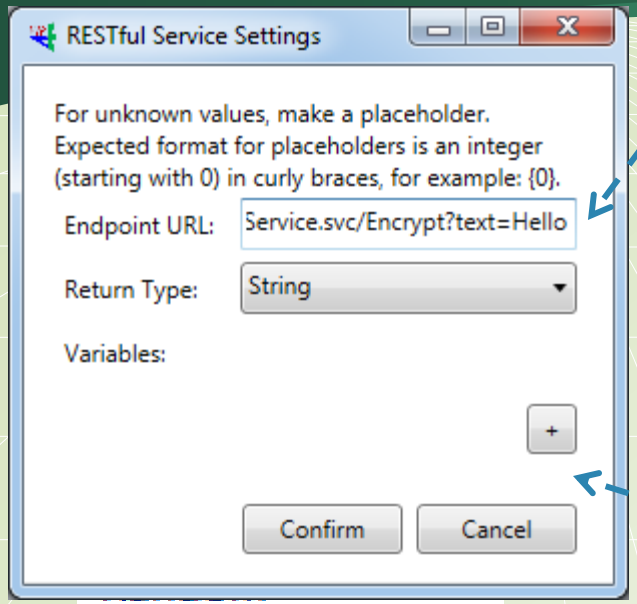




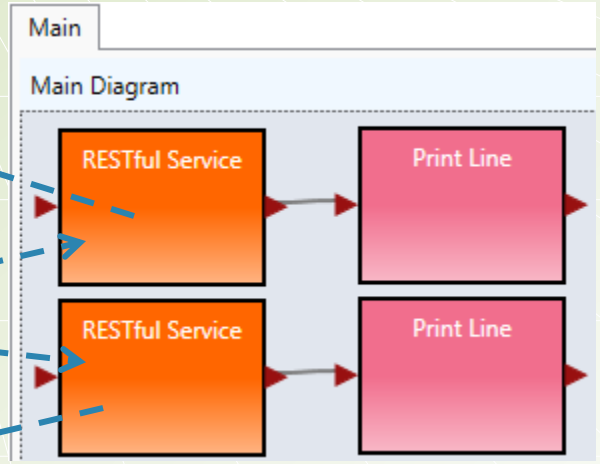
After a Custom Service is Created

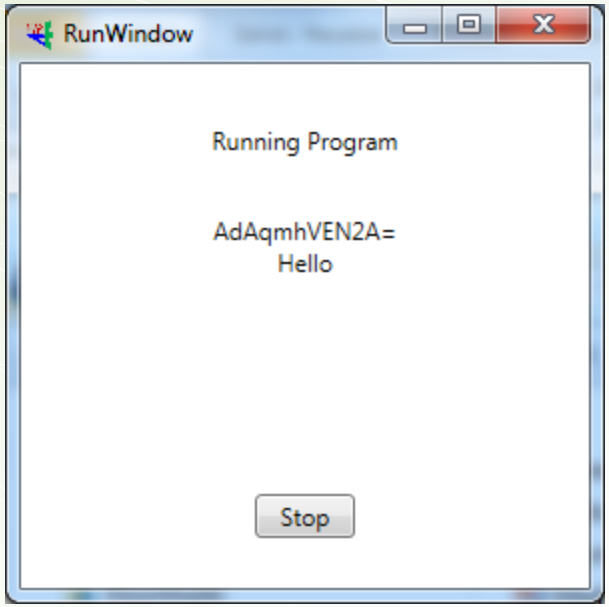
- When you “Export as a Service”, you can save the service anywhere you want.
- By default, it will be saved into the “CustomServices” folder in your VIPLE program folder.
- When VIPLE is started, all services will be imported into the VIPLE service list, where you find the other services like Print Line and Text to Speech.
- To delete (remove) a custom service, open the folder CustomServices and delete the file of the service. After you restart the service, the custom service will disappear from the service list.
- To share a service in another application, copy the service file into the CustomServices folder of another application.

Calling RESTful Services

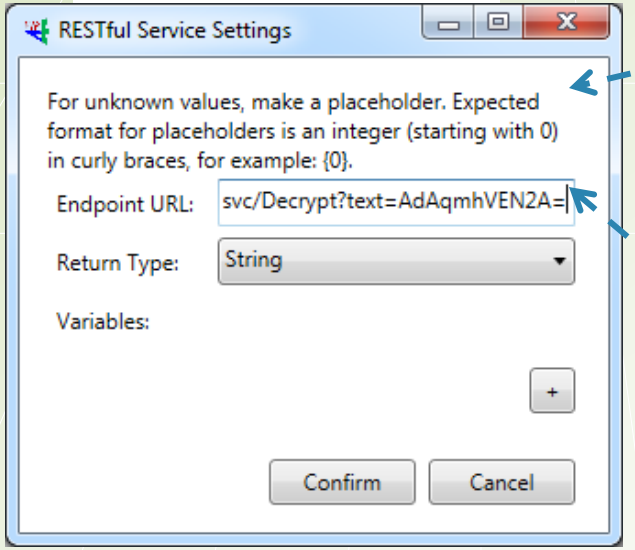


http://venus.eas.asu.edu/WSRepository/Services/EncryptionRest/Service.svc/Encrypt?text=Hello



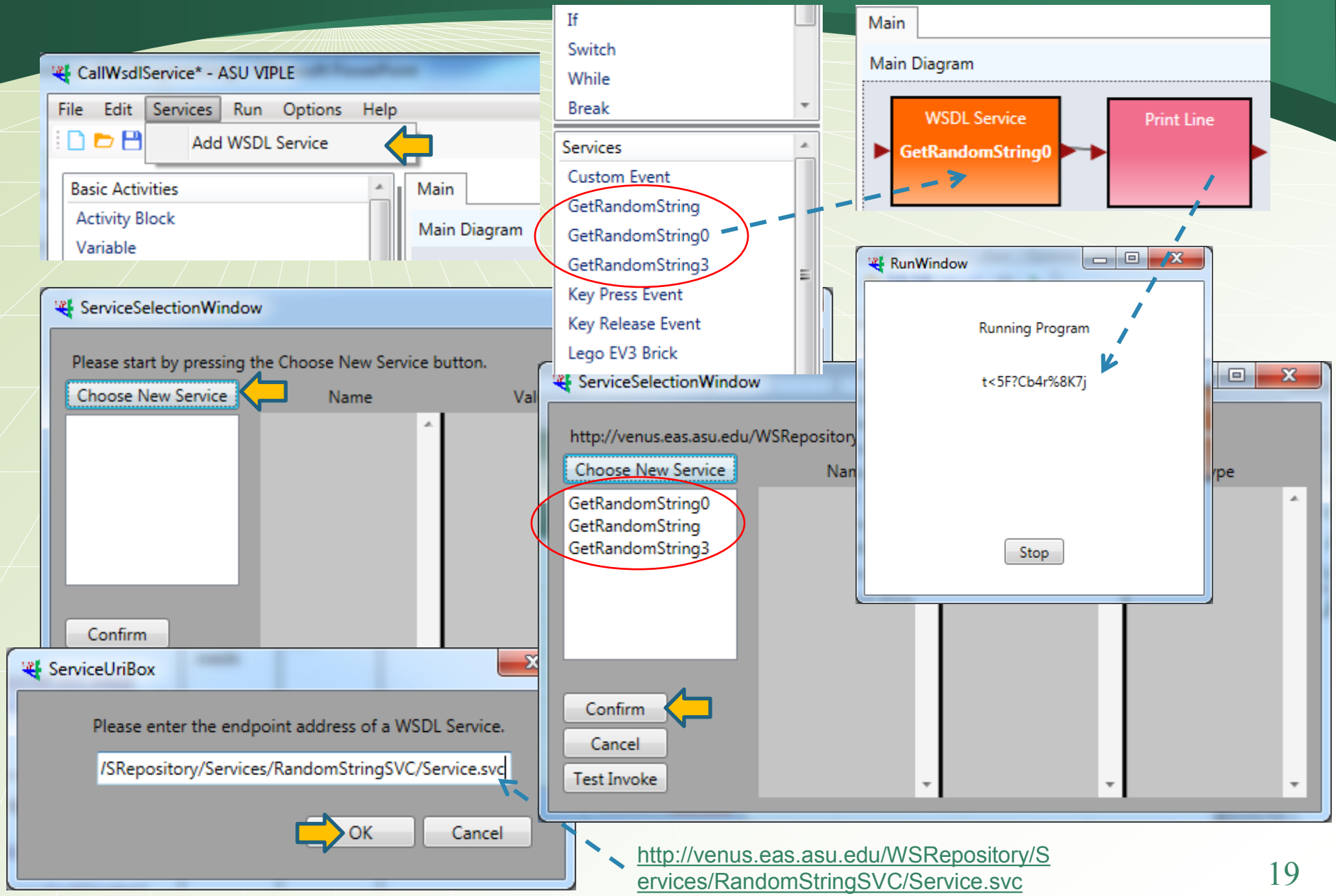


RESTful Service



http://venus.eas.asu.edu/WSRepository/Services/EncryptionRest/Service.svc/Decrypt?text=AdAqmhVEN2A=

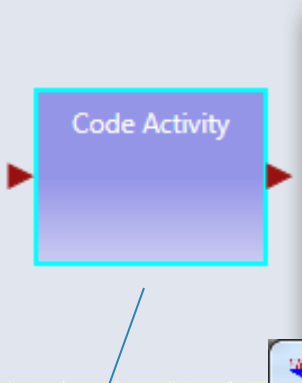
Calling a WSDL Service

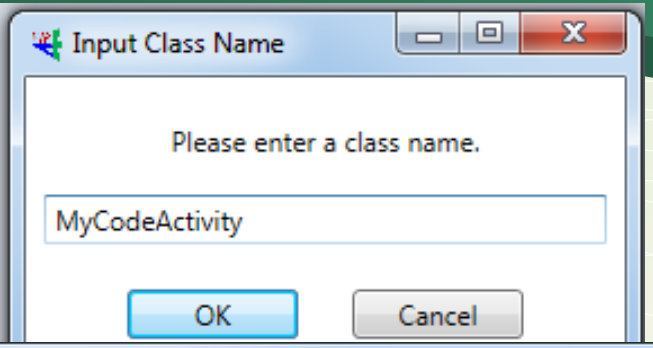


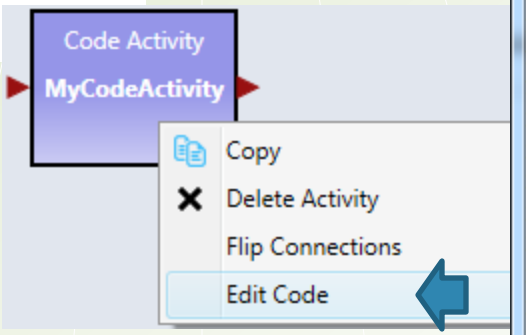
http://venus.eas.asu.edu/WSRepository/Services/RandomStringSVC/Service.svc

Code Activity: Wrap any C# Class into an Activity

- Code Activity
- Custom Event
- Key Press Event
- Key Release Event
- Print Line
- Random
- RESTful Service
- Simple Dialog
- Text to Speech
- Timer







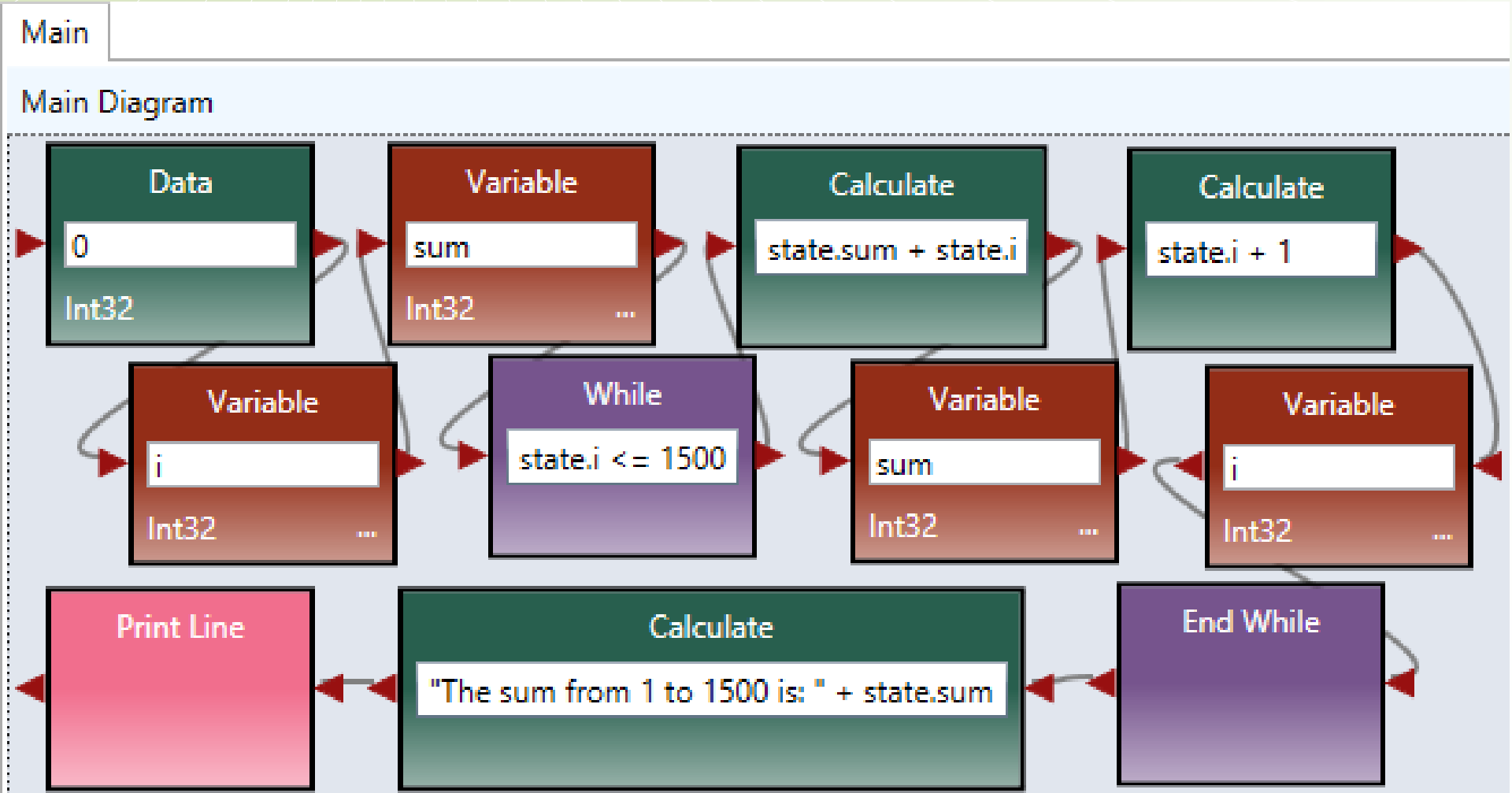
Code Editor

```
using System;

public class MyCodeActivity : CodeUtilities.CodeBase
{
    // Setting the return type up here allows you to use the "value"
    // keyword correctly in connected activities.
    public MyCodeActivity()
    {
        ReturnType = typeof(Double);
    }

    // To execute your code, you must override the Execute method.
    public override void Execute()
    {
        // Obtain the value of the input to this activity.
        // The type of this value will depend on what input you pass
        // to this activity.
        Int32 myInput = (Int32)Input.Value;

        // You can use the PrintLine method to print strings
        // to the run window during runtime.
        PrintLine("Hello from MyCodeActivity!");

        // You can pass output in a similar way.
        Output.Value = 3.5;
    }
}
```

Sequential vs. Parallel Computing

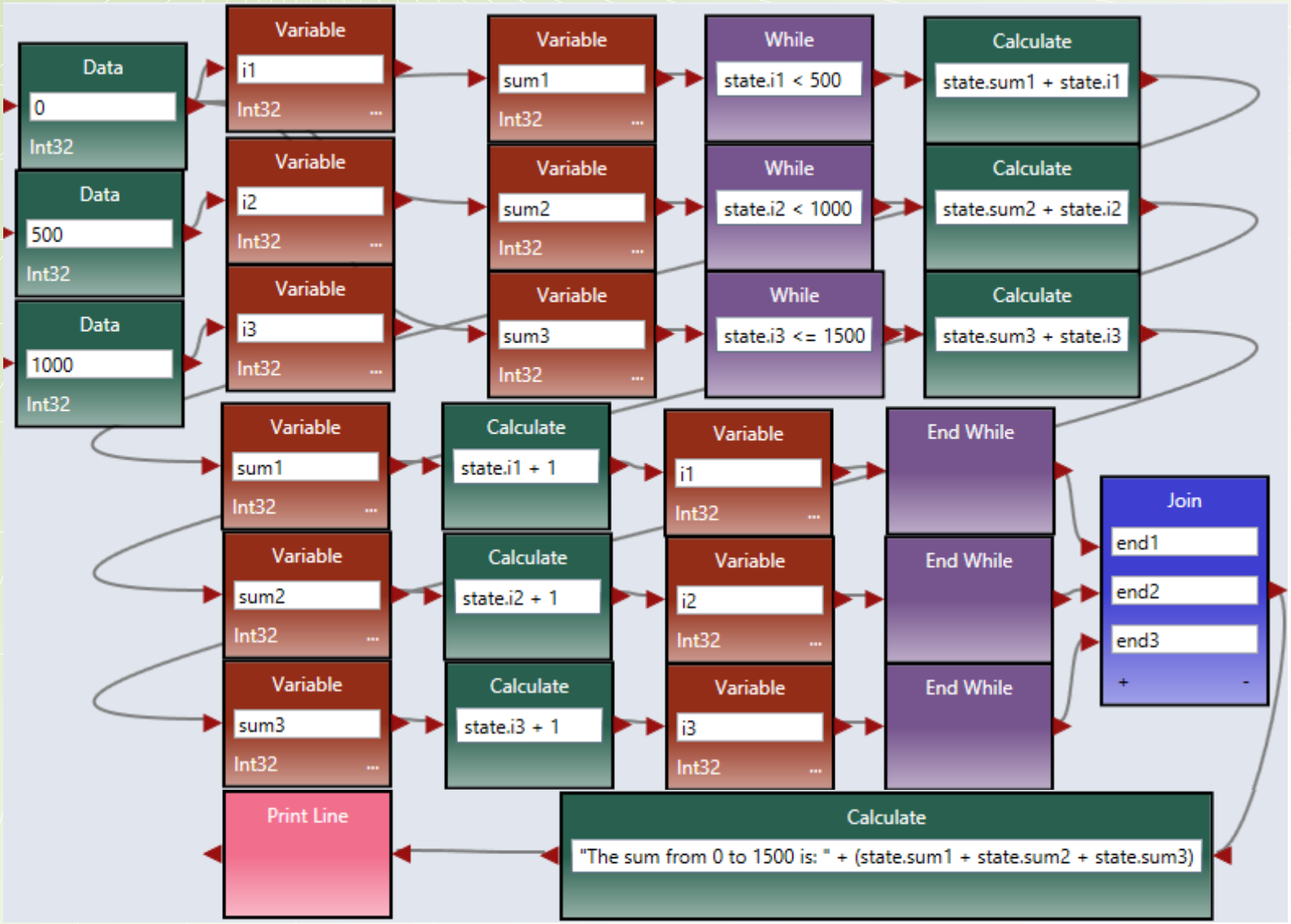
- Sequential version of adding many numbers

$$\text{sum} = \sum_{i=0}^{n} i$$



Parallel / Distributed Computing

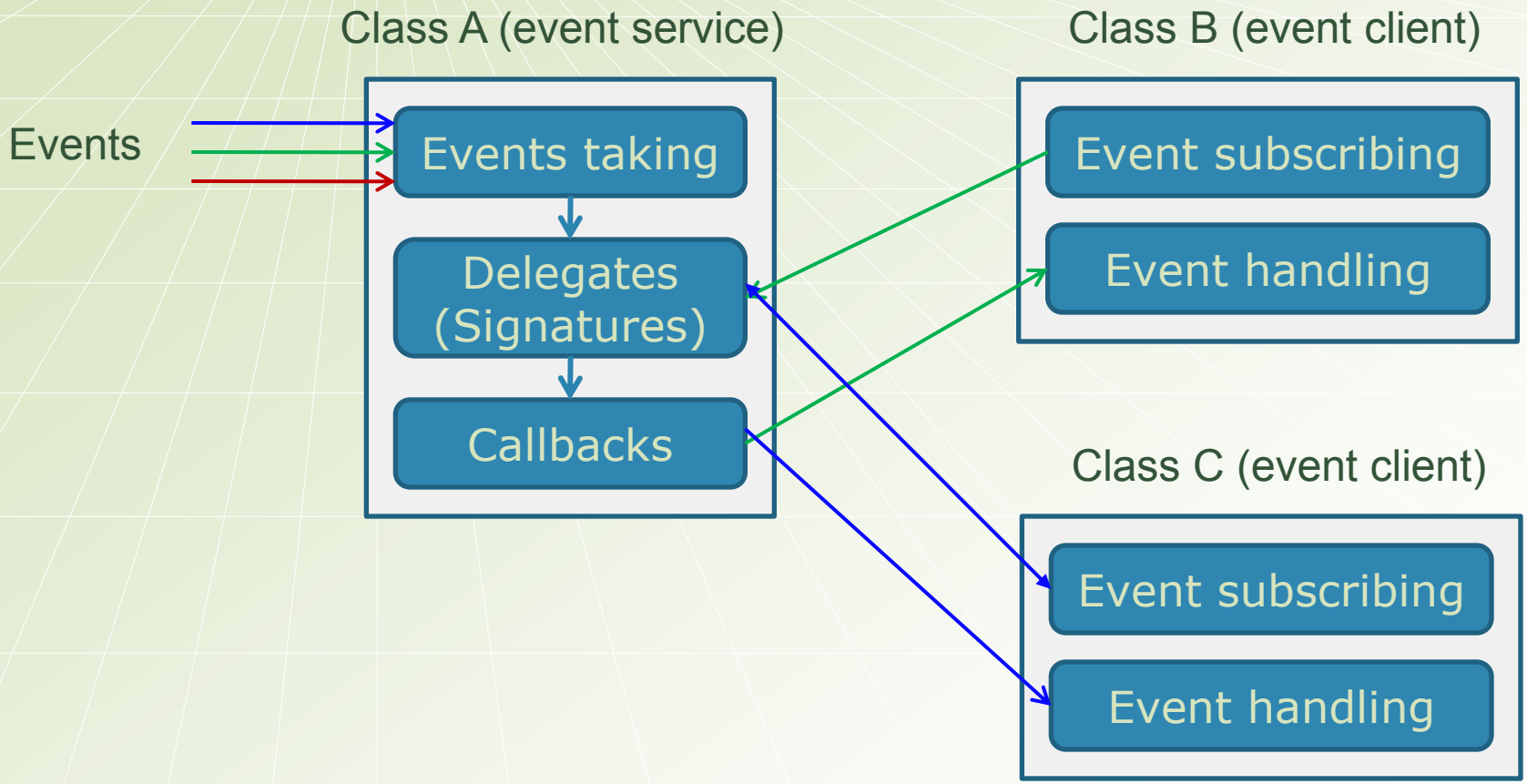
- Parallel version of adding many numbers



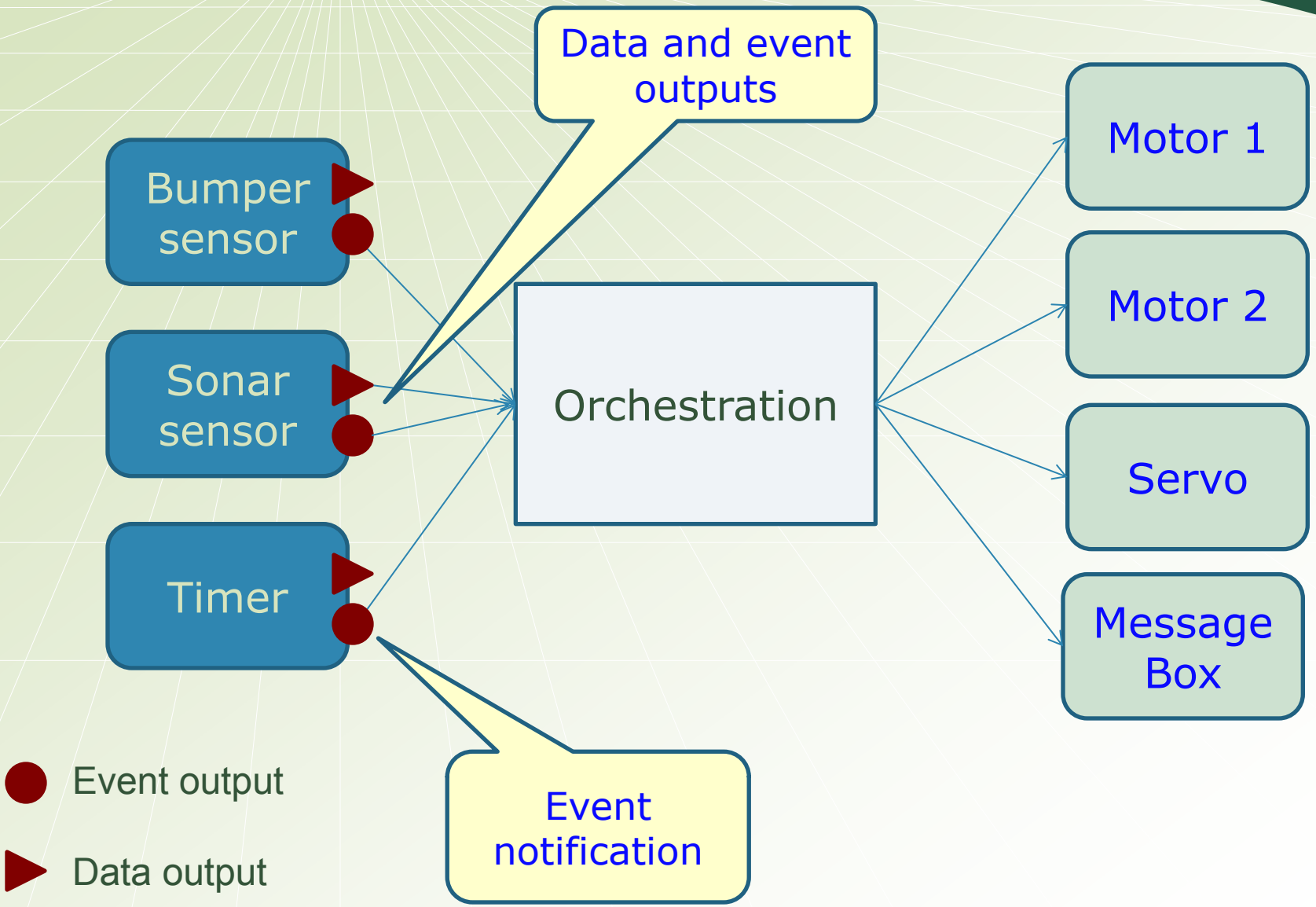
Events and Event Handling

- A common technique in distributed computing
  - XML validation and handling
  - Exceptions and handling
  - Mouse click and code processing the click
  - Sensory input arrived (touch sensor) and the action
  - A timer elapsed and the action
- Event-driven computing assumes there are multiple processors to handle events in parallel
- Event handling process
  - Class A publishes event delegates (signatures) for subscription;
  - Class B implements an event handler and subscribes to an event delegate by adding the handler name into the delegate;
  - When an event occurs in class A, class A will callback the handler in class B, which handles the event.

Events and Event Handling (Contd.)



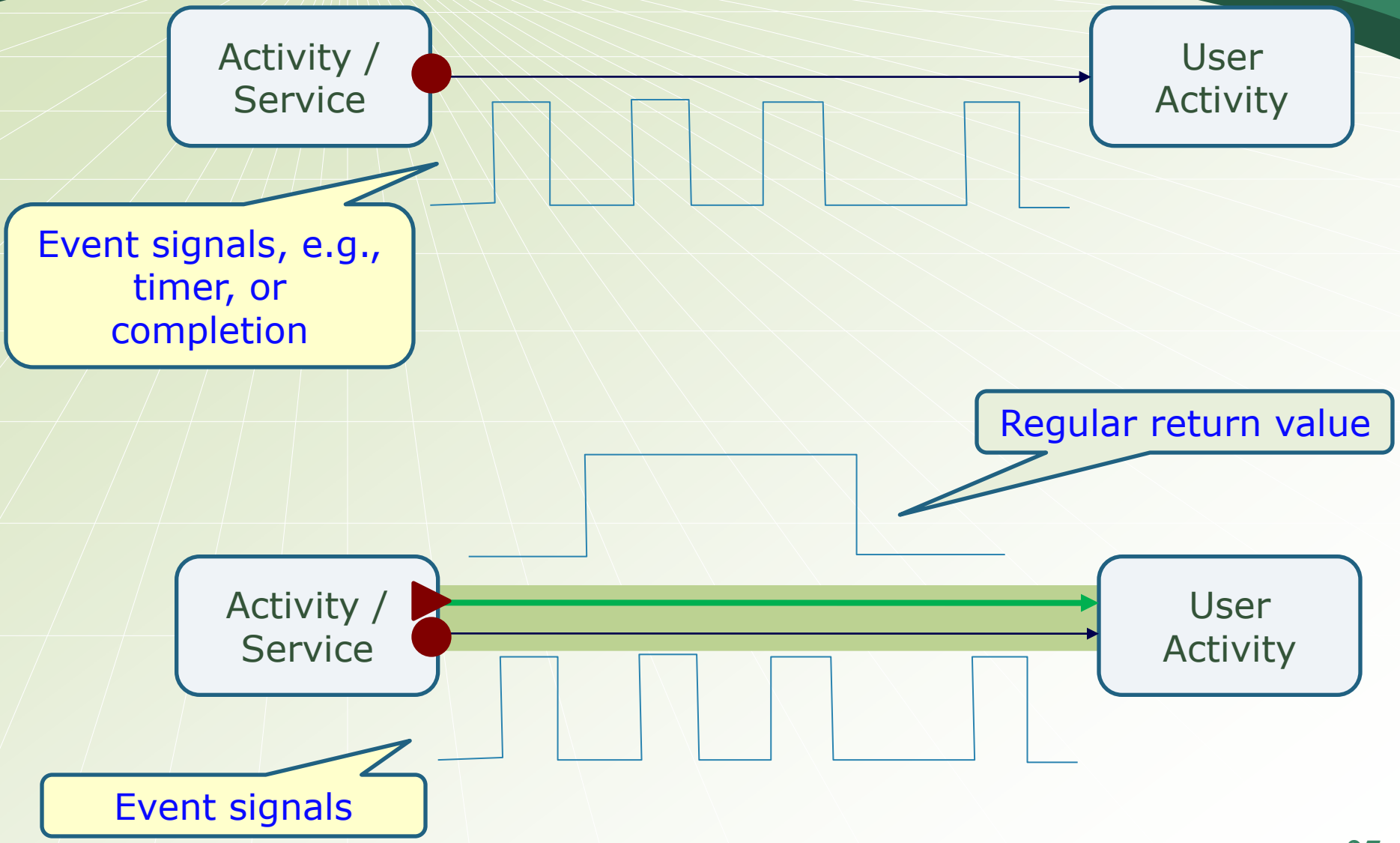
Concurrency and Events in Robotics Programming



Concurrency and Events in Robotics Programming

- Handling sensory inputs and controlling actuators must be dealt with concurrently, as otherwise sensor inputs can easily be ignored and actuators can get starved.
- Orchestration and composition should not be in control flow model. Event-driven model is a better way to handle such applications.
- Event notification can be sensible alone, or in combination with the return data

Event-Driven Notification

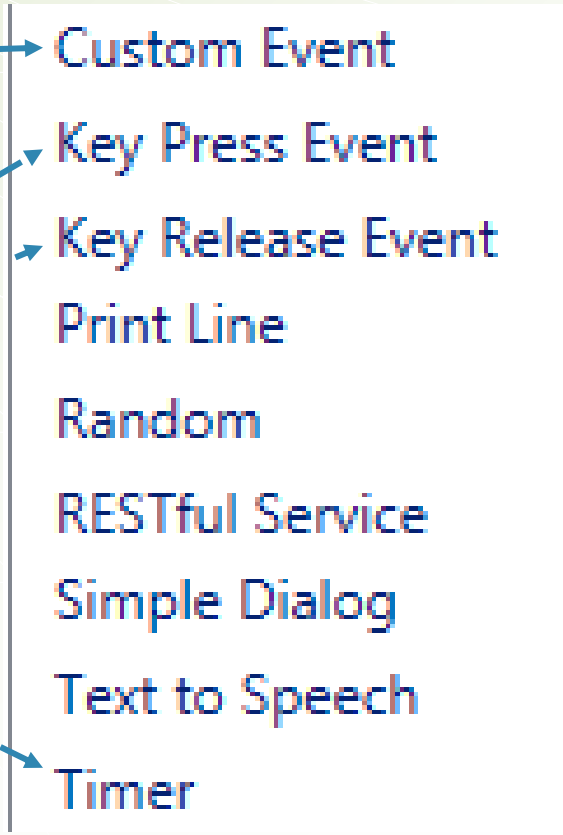


Event-Driven Programming

VIPLE supports two types of events

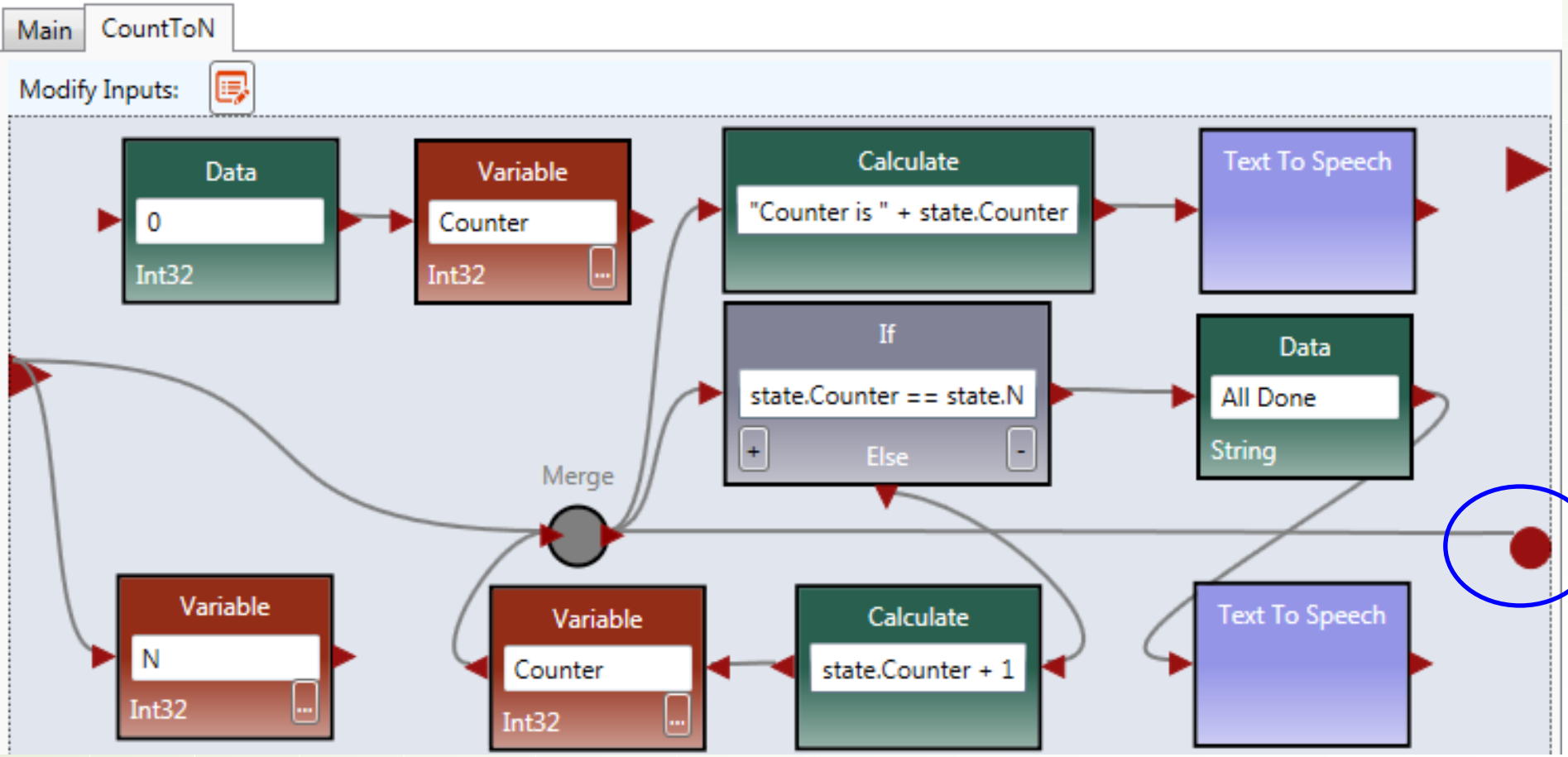
- Custom events: Allow programmers to define an event as an activity's output
- Built-in events: Predefined services in the VIPLE service list that generate events

General-purpose and event services



Event-Driven Programming: Custom Event

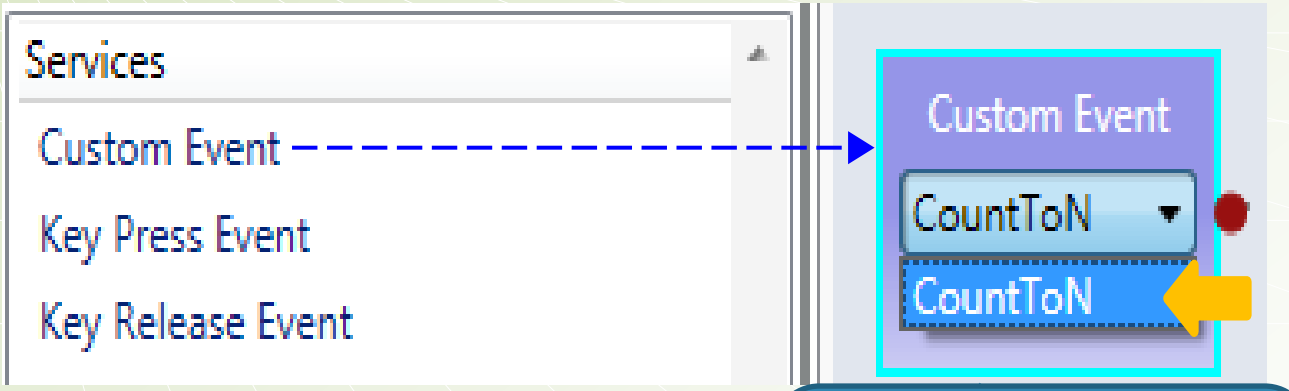
- Implementing the CountToN activity with event output



Event-Driven Programming: Custom Event

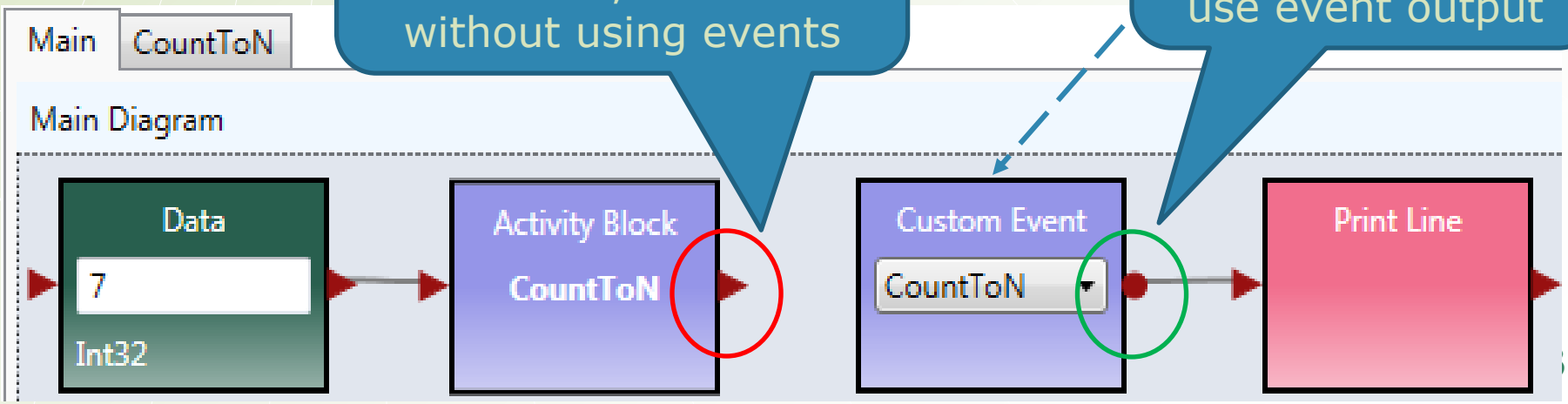
- Accessing the custom event

Custom events and built-in events



Counter generates different data in each iteration, it works without using events

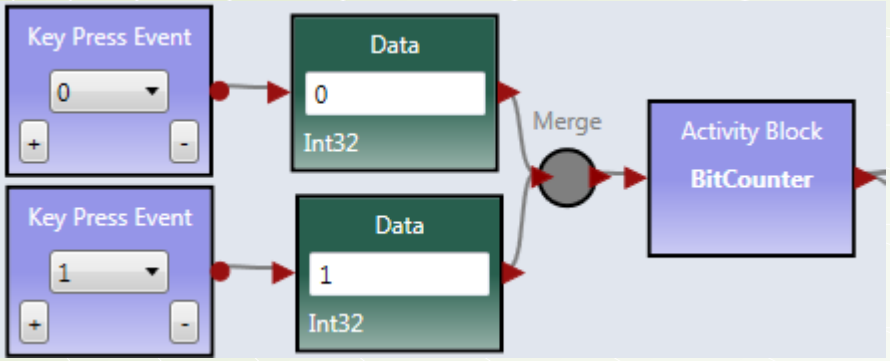
If an activity can generate the same data, we need to use event output

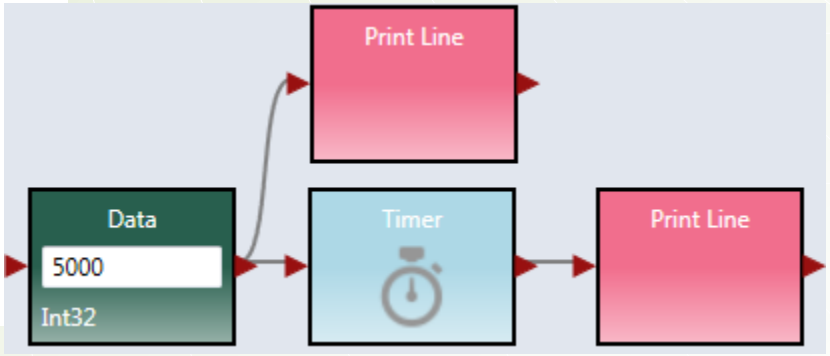


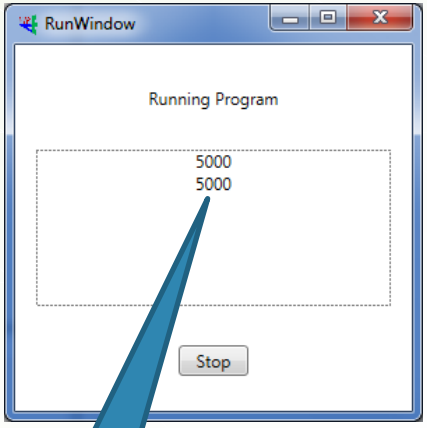
Event-Driven Programming: Key Press Event

General-purpose
and event services

- Code Activity
- Custom Event
- Key Press Event
- Key Release Event
- Print Line
- Random
- RESTful Service
- Simple Dialog
- Text to Speech
- Timer







Printed 5 second later

Parity Bit Generation Using Key Press Event

- An ASCII code consists of 7 bits of 0s or 1s.
- The 8th bit is often generated for parity checking:
  - If the first 7 bits has odd number of 1s, the 8th bit is 0, otherwise, it is 1, to keep the total number of 1s is an odd number.
- Write a VIPLE application to generate the odd-parity bit of an ASCII code. Example: