Flow-Based Programming: Why You Should Care Even If You Never Plan To Use It

OpenWest 2015
Samuel M Smith PhD
sam@ioflo.com
Reputation on the blockchain - it's time to get real.

worldtable.co
sam@worldtable.co
Computational reputation on the blockchain to modulate the internet of interactions.

Open source decentralized platform

openreputation.net
https://github.com/OpenReputation
Flow-Based Programming Framework

github.com/ioflo/ioflo

ioflo.com
Flow-Based Programming (FBP)

- Originated in 1970's by J.P. Morrison, contemporary with OOP & FP
- Relatively unknown but a lot of interest recently
- Think general-purpose data-flow programming
- Use it via a programming style, pattern, paradigm, or framework, not a language
- FBP is a simplifying unifying paradigm
- Distributed concurrent applications benefit most from FBP
- An FBP architecture may still be really useful even when not using an FBP Framework
- An FBP mindset may provide unique solution insights even when using OOP or FP
Flow-Based Programming Resources

Flow-Based Programming, 2nd Ed. May 14, 2010
http://www.jpaulmorrison.com/fbp/
http://www.jpaulmorrison.com/fbp/links_external.html
https://flowbasedprogramming.wordpress.com/article/flow-based-programming/

Port Automata/ Port Based Objects

Port Automata and the Algebra of Concurrent Processes, Steenstrup & Arbib 1982

SoftwareComponentsForRealTime, Stewart 2000
Flow-Based Programming Frameworks

Javascript
- NoFlo
- Flux
- constructables.es

Python
- ioflo
- PyF
- Pype

Other
- Expecco

Diagram:
- Port-based object (PBO)
- Configuration constants
- Variable input ports
- Variable output ports
- Resource ports for communication with sensors, actuators, and other subsystems

Table 1:

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pinvkin</td>
<td>Puma inverse</td>
<td>Compute inverse kinematics for a Puma 560 robot</td>
</tr>
<tr>
<td>pfwdkin</td>
<td>Puma forward</td>
<td>Compute forward kinematics for a Puma 560 robot</td>
</tr>
<tr>
<td>puma</td>
<td>Puma interface</td>
<td>Hardware-dependent interface to Puma 560 robot</td>
</tr>
<tr>
<td>cinterp</td>
<td>Cartesian trajectory</td>
<td>Given the current measured position and the interpolator desired final position, compute the intermediate kinematics parameters obtained during initialization of the module</td>
</tr>
</tbody>
</table>
| tball  | Trackball interface | A hardware-dependent interface to a six-degree model for different types of components is selected as an element of the PBO model, as depicted in Fig. 2. Resource ports are for modeling sensors, actuators, and other subsystems. PBO, resulting in an HD interface of data that is exchanged with I/O hardware. The algebraic model of a port involves arrows entering/leaving the top of the rectangle. Resource ports are shown as arrows entering/leaving the side of the rectangle. Configuration constants are drawn as a round-corner rectangle. Resource ports are implemented in a hardware-dependent manner, as the reads and writes of the I/O hardware are freely classified as generic components for use with specific hardware or applications. The resource ports are for modeling sensors, actuators, and other subsystems. The algebraic model of a port is an extension to objects to include sensors, actuators, and other subsystems. As stated by Wegner's distinction of object-oriented design, the term object does not imply that objects without inheritance are classified as objects. In C, the objects are implemented as ADTs. Communication with other modules is restricted. In object-oriented design, the ref-
More or less FBP

Java VM
Apache Storm
IBM InfoSphere Streams
Microsoft Azure Event Hubs

Python
ROS
Open Source Robotics Foundation

Other
MatLab
Simulink
Erlang
LabVIEW
What is Flow-Based Programming (FBP)

A behavior transforms its input(s) into its output(s)

Inputs may also be parameters that modify the transformation

A behavior transforms its inputs governed by its parameters into its outputs
What is Flow-Based Programming (FBP)

Outputs may be fed back as state to modify the transformation

A behavior transforms its inputs and past outputs as governed by its parameters into its outputs.
What is Flow-Based Programming (FBP)

FBP applications (programs) are networks of asynchronous behaviors which exchange data across externally defined connections.

Behaviors can be reconnected endlessly in different networks without any internal changes.
What is Flow-Based Programming (FBP)

- Programming is the declarative composition of networks of behaviors
- FBP = Data Flow Oriented + Component Oriented
- FBP = FP-ish + OOP-ish
FBP Variant with DataStore Substrate

From this

To this

- Adds configurability, observability, traceability, replayability
What is a Flow-Based Programming Framework

- Analogous to web application framework.

- Provides syntactic sugar and/or graphical editors to facilitate the composition and scheduling of behavior networks/graphs.

- Begins where many other distributed application architectures are striving to end up.

```plaintext
framer vehiclesim be active first vehicle_run
  frame vehicle_run
    do simulator motion uuv

framer mission be active first northleg
  frame northleg
    set elapsed to 20.0
    set heading to 0.0
    set depth to 5.0
    set speed to 2.5
    go next if elapsed >= goal

framer eastleg
  set heading to 90.0
  go next if elapsed >= goal

framer southleg
  set heading to 180.0
  go next if elapsed >= goal

framer westleg
  set heading to 270.0
  go next if elapsed >= goal

framer mission_stop
  bid stop vehiclesim
  bid stop autopilot
  bid stop me

framer autopilot be active first autopilot_run
  frame autopilot_run
    do controller pid speed
    do controller pid heading
    do controller pid depth
    do controller pid pitch
```
Bad Software Design: Software that fulfills its requirements but exhibits any or all of:

- **Rigidity**: Hard to change because each change affects other parts of the system.
- **Fragility**: Making a change causes other parts of the system to break.
- **Immobility**: Hard to disentangle in order to reuse in another application.

**DIP:**

- **HIGH LEVEL MODULES SHOULD NOT DEPEND UPON LOW LEVEL MODULES.**
- **BOTH SHOULD DEPEND UPON ABSTRACTIONS.**
- **ABSTRACTIONS SHOULD NOT DEPEND UPON DETAILS.**
- **DETAILS SHOULD DEPEND UPON ABSTRACTIONS.**

It's all about dependency management, duh !!!
Dependency Inversion Principle Transcendence

Bad Software Design: Software that fulfills its requirements but exhibits any or all of:

- **Rigidity**: Hard to change because each change affects other parts of the system.
- **Fragility**: Making a change causes other parts of the system to break.
- **Immobility**: Hard to disentangle in order to reuse in another application.

Flow-Based Programming transcends the DIP thusly:

- **THERE ARE NO MODULES, JUST COMPONENTS**
- **THERE ARE NO ABSTRACTIONS OR DETAILS JUST DATA**
- **COMPONENTS DEPEND ON DATA, NOT OTHER COMPONENTS**
One Dependency: DATA

Data Store Substrate
Replacement Independence

Behaviors (components) can be externally connected without any internal changes.

Composition of complex networks/graphs is conceptually simple

Because partitioning occurs intra-process/intra-host instead of inter-process/inter-host, distribution of behaviors across processor resources does not change behavior internals

Replacement Independence = Dependency Minimization

Replacement Independence = Flexibility, Robustness, Mobility
Complexity Management

Real Complexity = number of dependencies between elements of a software system

Apparent Complexity = number of dependencies that programmers must manage in order to make meaningful enhancements to software functionality

Perceived Risk = peril the programmer faces when attempting to add meaningful enhancements to software functionality.
Scalability

Higher levels of capability increase *real complexity* and may increase *apparent complexity*.

The principle limitation is programmer capacity *not* computational capacity.
Distributed Application Menagerie

- Pub/Sub Message Broker (Rabbit MQ, AMQP, ZeroMQ)
- Task Scheduler (Celery, BeanStalk)
- Key Value Store, Database (Redis, Couch, Mongo, River, ...)
- Resource Manager (Zookeeper)
- Data Flow Processor (Storm, Elastic, ...)
- Services (ReST web, WebRTC, Soap, ...)
Autonomous Underwater Vehicles
Autonomous Underwater Vehicles
Shipboard & Building Automation
Convenient Declarative Syntax (3rd Generation)
- Unified Scheduling and Execution
- Integrated Pub/Sub, Messaging,
- Observable/Traceable
- Nested Concurrent Contexts

all in one place
not block oriented

Declaration Sentence:  

<table>
<thead>
<tr>
<th>Verb</th>
<th>Object</th>
<th>Prepositional Phrases</th>
</tr>
</thead>
<tbody>
<tr>
<td>done</td>
<td></td>
<td></td>
</tr>
<tr>
<td>frame build in setup</td>
<td>set speed to 5</td>
<td></td>
</tr>
<tr>
<td>framer create at 0.5 be active first build</td>
<td>[adjectives ...] kind</td>
<td></td>
</tr>
</tbody>
</table>

Verb Object:  
do salt eventer
Hierarchical State Machine

Hierarchical State Machine
Relatively Low Apparent Complexity

Flat State Machine
Relatively High Apparent Complexity

Hierarchical State Machine Tree
Transition = Change in Framing Context

```plaintext
go foobar if elapsed >= goal
```
Dichotomous Priorities

Transitions
Change Context, Propriety
Priority is Top-Down

Hierarchical State Machine Tree

Actions
Determine Contextual Behavior, Specificity
Priority is Bottom-Up
Reliability

Reliable Services Envelope
High Level Frames - Programmed by Experts

Failure Checks → Failure Handling

Common Generic
Programmed by Experts
Action

Mission Specific
Programmed by Non-Experts
Action
Action
Action
**Framing**
- concurrency
- Framers
- Auxiliary Framers
- Master/Slave Framers
- Nested Frames

**Actioning**
- components
- FloScript Verbs
- Ioflo library Python Actions/Behaviors for do verb objects
- Custom Python coded do verb objects
- Custom coded Python C Extensions for do verb objects
Universal Name-Spaced Pub/Sub Interface for Intra application communication

*high replacement independence*

Addressing Modes:

- **direct**
  - put name sam eyes blue into .person.detail
  - set speed by blue in myapp.setup
  - put name sam eyes blue into detail of frame start

- **indirect-absolute**

- **indirect-relative-implied**

- **indirect-relative-root**

- **indirect-relative-frame**

- **indirect-relative-framer**

  - put true into good of framer
RAET Reliable Asynchronous Event Transport

https://github.com/RaetProtocol/raet

- RAET
  - Micro threaded architecture with non-blocking I/O
  - Micro threaded Event Pub/Sub separate from socket based transport layer
  - UDP sockets
  - Better observability and management of performance under load
  - Transactions
  - Unix domain sockets for interprocess communications
References

http://ioflo.com
https://github.com/ioflo
https://github.com/ioflo/ioflo_manuals

http://www.jpaulmorrison.com/fbp/
http://www.jpaulmorrison.com/fbp/links_external.html
https://flowbasedprogramming.wordpress.com/article/flow-based-programming/
http://www.amazon.com/Flow-Based-Programming-2nd-Application-Development/dp/1451542321/ref=sr_1_1?ie=UTF8&qid=1427910581&sr=8-1&keywords=flow+based+programming

http://wiki.ros.org/ROS/Tutorials
Backup Slides
$ pip install ioflo

$ ioflo -h


Runs ioflo. Example: ioflo -f filename -p period -v terse -r -h -b 'mybehaviors.py'

optional arguments:
    -h, --help            show this help message and exit
    -v VERBOSE, --verbose VERBOSE
                           Verbosity level.
    -p PERIOD, --period PERIOD
                           Period per skedder run in seconds.
    -r, --realtime       Run skedder at realtime.
    -V, --version        Prints out version of ioflo.
    -n NAME, --name NAME  Skedder name.
    -f FILENAME, --filename FILENAME
                           File path to FloScript file.
    -b [BEHAVIORS [BEHAVIORS ...]], --behaviors [BEHAVIORS [BEHAVIORS ...]]
                           Module name strings to external behavior packages.
    -U USERNAME, --username USERNAME
                           Username.
    -P PASSWORD, --password PASSWORD
                           Password.
house box1
  framer vehiclesim be active first vehicle_run
    frame vehicle_run
      do simulator motion uuv

  framer mission be active first northleg
    frame northleg
      set elapsed to 20.0
      set heading to 0.0
      set depth to 5.0
      set speed to 2.5
      go next if elapsed >= goal
      frame eastleg
        set heading to 90.0
        go next if elapsed >= goal
      frame southleg
        set heading to 180.0
        go next if elapsed >= goal
      frame westleg
        set heading to 270.0
        go next if elapsed >= goal
      frame mission_stop
        bid stop vehiclesim
        bid stop autopilot
        bid stop me

  framer autopilot be active first autopilot_run
    frame autopilot_run
      do controller pid speed
      do controller pid heading
      do controller pid depth
      do controller pid pitch

$ ioflo -v terse -f box1.flo

Starting mission from file box1.flo...
Starting Framer vehiclesim ...
To: vehiclesim<<vehicle_run> at 0.0
Starting Framer mission ...
To: mission<<northleg> at 0.0
Starting Framer autopilot ...
To: autopilot<<autopilot_run> at 0.0
To: mission<<eastleg> at 20.0 Via: northleg (go next if elapsed >= goal)
  From: <northleg> after 20.000
To: mission<<southleg> at 40.0 Via: eastleg (go next if elapsed >= goal)
  From: <eastleg> after 20.000
To: mission<<westleg> at 60.0 Via: southleg (go next if elapsed >= goal)
  From: <southleg> after 20.000
To: mission<<mission_stop> at 80.0 Via: westleg (go next if elapsed >= goal)
  From: <westleg> after 20.000
Stopping autopilot in autopilot_run at 80.000
Stopping vehiclesim in vehicle_run at 80.125
Stopping mission in mission_stop at 80.125
No running or started taskers. Shutting down skedder ...
Total elapsed real time = 0.2099
Aborting all ready taskers ...
  Aborting vehiclesim at 80.125
  Tasker 'vehiclesim' aborted
  Aborting mission at 80.125
  Tasker 'mission' aborted
  Aborting autopilot at 80.125
  Tasker 'autopilot' aborted
Intelligent Autonomy-Autonomic-Automation Programming Framework

Open Source: MIT License - ioflo.com - github.com/ioflo

Automation Operating System

Micro Threaded Concurrent Execution Engine With Non Blocking I/O

Automated Reasoning Engine

Flow Based Programming Framework

Hierarchical Action Framework

Discrete Event Simulation Framework

Dependency Injection Framework

Universal PubSub

FloScript: DSL for Convenient Configuration

Machine Learning - Machine Intelligence Infrastructure