## **REACT**

A Robot Programming Language

Sergio Benitez

### **Possible Solutions**

- Diverse Audience
- => Layered Power
- Many Possible Applications
- => Powerful Primitives

Complexity

- => Abstraction, Layering
- Fragile Robots/Conditions
- => Enforceable Constraints



# Challenges

- Diverse Audience
- Many Possible Applications
- Complexity
- Fragile Robots & Conditions



Power Complex

REACT:

**E**vent-Driven

**A**synchronous

Concurrent

Turing-complete



```
context Main {
    target = 3
    active walk_speed = Sonar.distance - target

whenever tooFar (Sonar.distance > target + .1) {
    Robot.walk! withSpeed:walk_speed
}

when justRight (target + .1 >= Sonar.distance >= target - .1) {
    Robot.stop!
    Main.end!
}

whenever tooClose (Sonar.distance < target - .1) {
    Robot.walk! withSpeed:walk_speed
}

context Sonar {
    public active distance = getSonarDistance()
}</pre>
```

```
context Main {
  target = 3
  active walk_speed = Sonar.distance - target

whenever tooFar (Sonar.distance > target + .1) {
  Robot.walk! withSpeed:walk_speed
}

when justRight (target + .1 >= Sonar.distance >= target - .1) {
  Robot.stop!
  Main.end!
}

whenever tooClose (Sonar.distance < target - .1) {
  Robot.walk! withSpeed:walk_speed
}
}

context Sonar {
  public active distance = getSonarDistance()
}</pre>
```

```
context Main {
  target = 3
  active walk_speed = Sonar.distance    target

whenever tooFar (Sonar.distance > target + .1) {
    Robot.walk! withSpeed:walk_speed
}

when justRight (target + .1 >= Sonar.distance = target - .1) {
    Robot.stop!
    Main.end!
}

whenever tooClose (Sonar.distance < target - .1) {
    Robot.walk! withSpeed:walk_speed
}

context Sonar {
    public active distance = getSonarDistance()
}</pre>
```

```
context Main {
  target = 3
  active walk speed = Sonar.distance - target

whenever tooFar (Sonar.distance > target + .1) {
   Robot.walk! withSpeed:walk_speed
}

when justRight (target + 1 = Sonar.distance >= target .1) {
  Robot.stop!
  Main.end!
}

whenever tooClose (Sonar.distance < target .1) {
  Robot.walk! withSpeed:walk_speed
}

context Sonar {
  public active distance - getSonarDistance()</pre>
```

```
context Robot {
   action walk! withSpeed.speed {
    walkRobot(speed) // C call
}

action turn! toDirection:dir withVelocity:vel:20 turnRadius:rad:90 {
   constrain dir (dir == -1 || dir == 1)
   constrain vel (0 <= vel <= 100)
   constrain rad (-90 <= rad <= 90)

   turnRobot(dir, vel, rad) // Another C call
}

action stop! {
   stopRobot() // One more C call
   trigger Robot.stopped // on (Robot:stopped) { //do_this }
}
</pre>
```

```
context Robot {
  action walk! withSpeed:speed {
    walkRobot(speed) // C call
}

action turn! toDirection:dir withVelocity:vel:20 turnRadius:rad:90 {
    constrain dir (dir == -1 || dir == 1)
    constrain vel (0 <= vel <= 100)
    constrain rad (-90 <= rad <= 90)

    turnRobot(dir, vel, rad) // Another C call
}

action stop! {
    stopRobot() // One more C call
    trigger Robot:stopped // on (Robot:stopped) { //do_this }
}</pre>
```

```
context Robot {
  action walk! withSpeed:speed {
    walkRobot(speed) // C call
}

action turn! toDirection:dir withVelocity:vel:20 turnRadius:rad:90 {
    constrain dir (dir == 1 || dir == 1)
    constrain vel (0 <= vel == 100)
    constrain rad ( .90 <= rad <= 90)

    turnRobot(dir, vel, rad) // Another C call
}

action stop! {
    stopRobot() // One more C call
    trigger Robot:stopped // on (Robot:stopped) { //do this }
}</pre>
```

```
context Main {
  lastX = 0
  lastY = 0

action start! {
    log "Starting!"
  }

whenever (Sensor.xPosition != lastX || Sensor.yPosition != lastY) {
    Head.look! toX:Sensor.xPosition toY: Sensor.yPosition
    lastX = Sensor.xPosition
    lastY = Sensor.yPosition
}
```

# **Key Properties**

- √ Enforceable Constraints (Constraints)
- √ Layering (Contexts)
- ✓ Primitives (Actions, Events)
- √ Abstraction, Layering (Libraries)
- √ Reactive (Event-Driven)

```
context Main {
  lastX = 0
  lastY = 0

action start! {
   log "Starting!"
}

whenever (Sensor.xPosition != lastX || Sensor.yPosition != lastY) {
   Head.look! toX:Sensor.xPosition toY: Sensor.yPosition
   lastX = Sensor.xPosition
   lastY = Sensor.yPosition
}
```

## **Extensions & Issues**

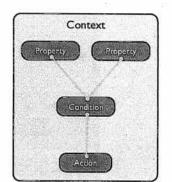
- Concept of time hidden initially
  - Can expose via Timer context
  - Extend language with RTC concepts
- Minimum microprocessor requirement
  - Need sufficient power
- Users may fall back to C too often
  - Okay or extend language?

## Risks

- Too Simple
- Too Complex
- Too "Different"
- Incomplete Controller Library

# Graphical Version

- Unique UI Elements
  - Properties
  - Conditions
  - Actions
- · "Linking" mechanism
- Contexts



# Comparison

- Text Based:
  - RoboForth
  - RobotC
  - WPI Robotics Library
- Graphical:
  - RoboLab
  - EasyC
  - NXT-G

ROBOTO





## RobotC

## RobotC

### RobotC

#### RobotC

#### RobotC

```
task main() {
  bMotorReflected[port2] = 1;

SensorValue[leftEncoder] = 0;
SensorValue[rightEncoder] = 0;

while (SensorValue[sonarSensor] > 3 || SensorValue[sonarSensor] < 0) {
  moveStraight(SensorValue[sonarSensor] - 3);
}
</pre>
```

#### **REACT**

```
context Main {
  whenever (Sonar.distance > 3 || Sonar.invalid) {
    Robot.walk! withSpeed: Sonar.distance - 3
  }
}
```

### RobotC Pros/Cons

- Pros:
  - Extremely powerful (C backbone)
- Cons:
  - · Requires users to understand C, timing
  - Steep learning curve
  - Polling mechanism, impossible to react

#### Possible Solutions

Fragile Robots/Conditions => Enforce

=> Enforceable Constraints

Diverse Audience

=> Layered Power

Many Possible Applications

=> Powerful Primitives

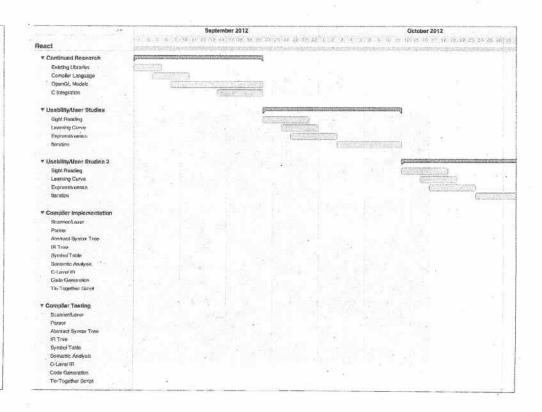
Unthinkable Complexity

=> Abstraction, Layering

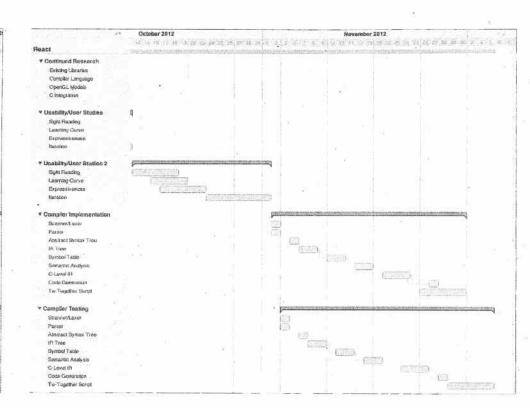


# Language Comparison

Language	Easy To Use	Resource Constraints	Temporal Constraints	Event Driven	Async	00
RobotC	*	***	1	*	*	*
RForth		**	<b>V</b>	**	*	*
WPI Lib	82	**	- V	*	*	
REACT	1		***		<b>V</b>	



38			October 2012	Desertion to a s	No.	November
	178 78 70 04 77 0454	PUBLICATION SCHOOL	2015 W-17 H 16 W 16	186 to 36189 to	SAVE A PROPERTY OF	Car William Conservation
Teact	PERSONAL PROPERTY.	market in the second second	o Tall with engage of on		WWW.Walley	ATTEMPORAL STATE OF THE STATE O
* Continued Research	1					
Existing Libraries						
Comoler Language						
OpenGL Models	3				13 11	
G integration						
▼ Usability/User Studies	p-	ALSO AND DESCRIPTION OF THE PARTY OF THE PAR	3			
Eligist Processor						
Learning Curve	EB (2011)					11 22
Expressivenacy	E CONTROL OF THE PARTY OF THE P					
feeration						
			The same of the sa		and the	
▼ Usability/Usor Studies 2					4	
Eight Reading			( Control of the cont			
Learning Curve Expressiveness			ARTHURSCOME.			
torpressiveness heration			1.05 EVENT - CO.			
nerdade				Harmon	-	
V Compiler Implementation					AUGMATINGSTATION	Armonia infrascriptura
Scanner/Luner						
Pensor	- 1	1.7			277	
Acetruct Byrqua Tree					100	
Pi Tren						0=0
Symbal Table						10000
Semetic Analysis						1000
G-Leval IR						
Code Generation						
Tay Yogother Scrot						
* Compiler Testing					Ç.	
ScurnorA over				0	100	
Parsin						
Abstruct Syntax Tree					6	
IR Tree						
Symbal Table						
Sementic Analysis						
O-Level IFI						
Cado Generation						
Tie-Tagother Scripe						



November 2012 Y Continued Research Existing Ubraries Compiler Language OpenGL Models C Integration ▼ Usability/User Studies Sight Prending Leaning Curva Expressiones Districtures ▼ Uanbility/User Studies 2 Sight Reading Learning Curve Expressiveness Iteration ▼ Compiler implementation Scanner/Loxer Parner Abstruct System Trees DE Troo Symbol Table Someron: Analysis Code Consultion The Togother Sergit · Y Compiler Teating Abstract Syntax Tree IR Tree Symbol Table Comunitie Analysis C-Level IN Code Generation Tie-Together Script

REACT:
Event-Driven
Asynchronous
Concurrent
Turing-complete



4			×	
(2)				
E 180 G				
	a w		.*	
1 10				
21	70		*	
	30 a a a a a a a a a a a a a a a a a a a			
	¥ _ g		*/	
		40*	( <b>*</b> 8)	
	5) (4)		(8)	
			(P)	
	, ed s			
	51			
		34.09 0	1	
		8		
		4	X = 8 "	
	20 20 20 20 20 20 20 20 20 20 20 20 20 2			
		348 21		
		9		
	At .			
	5 N			
	C S S S S S S S S S S S S S S S S S S S	(4)		
	8 0		0 (40)	
74				
	5 8			1
				# ¥
	× ·			
		/		2 0
	55 類	1		i i
	1/2			= _
			×	
				12
	0.			
	W			
	31 yr	R (4)		
Er Si			34	15
			9	
			34	