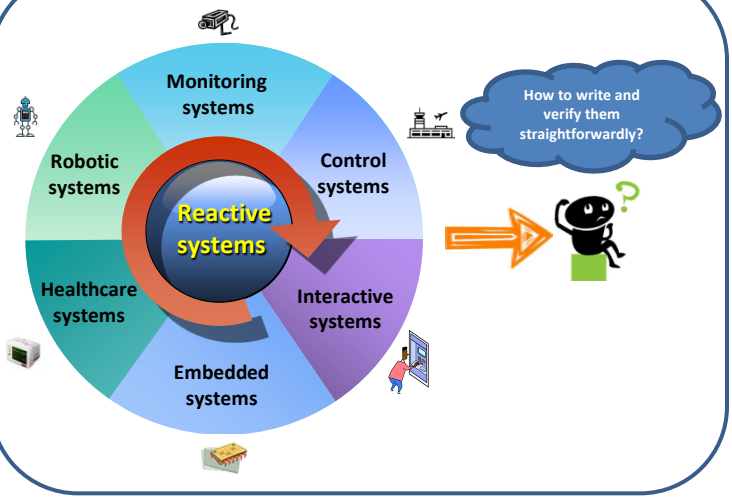
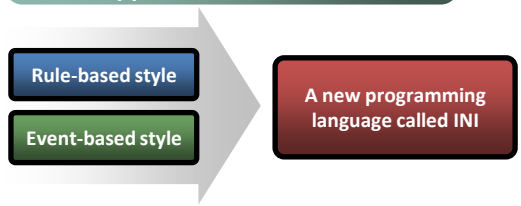
Combining Rule-based and Event-based Programming Paradigms for the Development of Reactive Systems

Truong-Giang Le
LISITE-ISEP, 28 rue Notre-Dame des Champs, 75006 Paris, France
Email: le-truong.giang@isep.fr

1. Motivation



2. Our Approach



Major features

- Rules and events can be defined independently or in combination.
- Programmers may write user-defined events in Java or in C/C++, then integrate them into INI programs.
- Events may run in parallel either asynchronously or synchronously.
- Events can be reconfigured at runtime to change their behaviors.

3. Examples

A simple HTTP server

```
function main() {
  @init() {
    start_http_server(8080)
  }
}
function start_http_server(port) {
  @init() {
    s = socket_server(port)
    clear(c)
    println("Server started on " +
     port)
  }
  s {
    c = socket_accept(s)
  }
  @update[variable = c](oldc, newc) {
    // Handle HTTP requests here
    client = socket_address(c)
    ...
  }
}
```

A prototype for an M2M gateway

```
function main() {
  @init() {
    dataFile = file("faceData.csv")
  }
  $(e) f:@faceDetect[period = 100](point1,
   point2) {
    case {
      !file_exists(dataFile) {
        create_file(dataFile)
      }
    }
    fwriteln(dataFile, to_string(time()) + ","
     + point1 + "," + point2)
  }
  $(f) e:@every[time = 5000]() {
    upload_ftp("host", "username", "password",
     dataFile, to_string(time()) +
     "dataUpload.csv")
    delete_file(dataFile)
  }
}
```

Syntax:
Rule: <Condition> {<Action>}
Event: $(<List of synchronized events>) id:@event[<Input parameters>](<Output parameters>) {<Action>}

4. Type System in INI

Common types

- Number types (Double, Float, Long, Int, Byte), Char type, String type, and Map types (List, Set).

User-defined types

```
type Company = [name:String, numOfEmployee:Int]
function main() {
  @init() {
    c = Company[name = "XYZABC",numOfEmployee = 100]
    println("The " + c.name + " company has " +
     c.numOfEmployee + " employees.")
  }
}
```

Type inference and checking engine

- No need to declare any type explicitly.
- Type conflicts are prohibited.

5. Model Checking INI Programs

INI can be converted to Promela for model checking with SPIN. Possible applications are:
- Detecting infinite loops and unreachable code.
- Verifying properties expressed in Linear Temporal Logic formulas.

Example: Detecting unreachable code

```
function main() {
  @init() {
    v=1
  }
  v < 5 {
    v++
  }
  v == 6 {
    v = v+2
  }
}
```

```
int v = 1
proctype main() {
  do
    ::v < 5 -> v++;
    ::v == 6 -> v = v+2;
    ::else -> break;
  od;
}
init {
  run main();
}
```

Verification result

```
...
unreached in proctype main
  UnreachableCode.pml:5, state 4,
   "v = (v+2)"
  (1 of 10 states)
unreached in init
  (0 of 2 states)
...
```

Project: MCUBE
Sponsors: SYSTEMATIC PARIS REGION SYSTEMS & ICT CLUSTER, l'Europe s'engage en Ile-de-France avec le FEDER

Reference:
https://sites.google.com/site/inilanguage