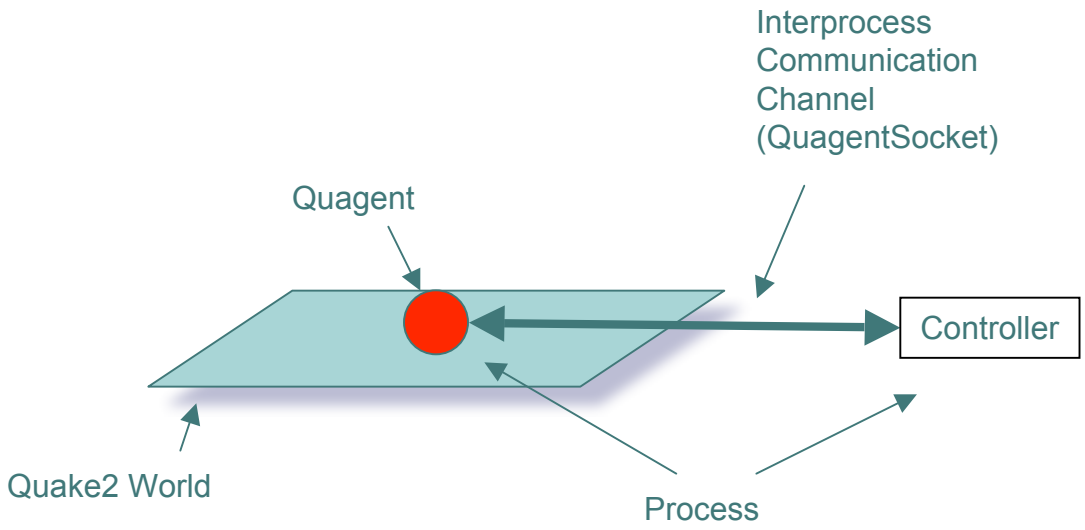
Interprocess Communication

- Why is it so tricky to program Quagents?
  - The body is represented by one process
  - The controller by another process
  - Both processes communicate with each other by passing messages
  - The really tricky part is that these messages are asynchronous!

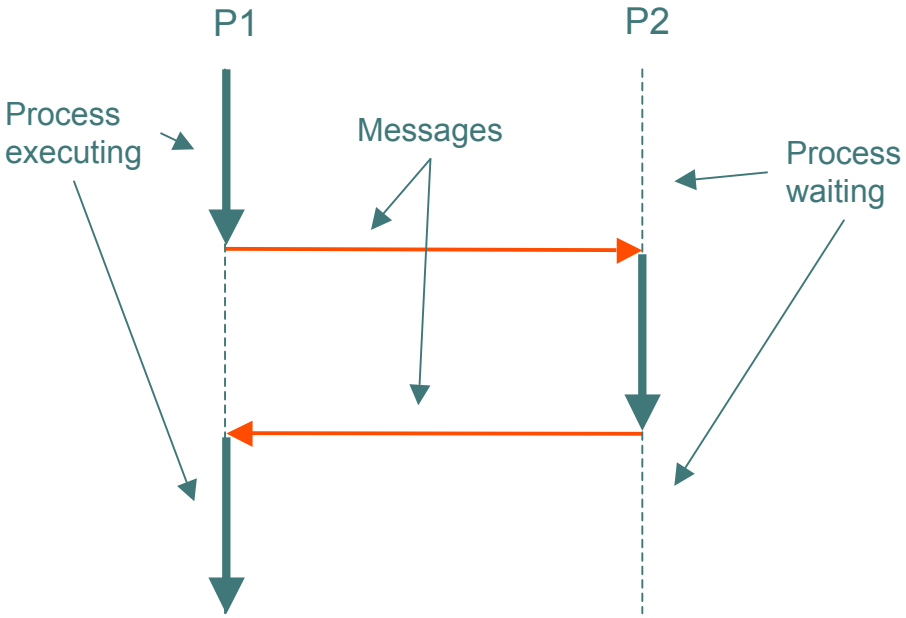
IPC ≡ Interprocess Communication

Interprocess Communication



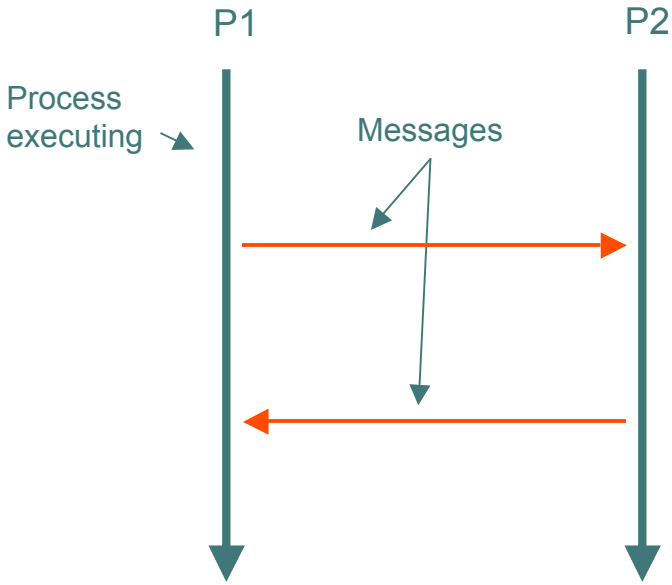
Interprocess Communication

Synchronous Communication



Hand Shaking

Asynchronous Communication



No Waiting!

Interprocess Communication

- Asynchronous communication is more natural in our setting
- Consider the alternatives:
  - the brain stops working while body is walking
  - the body stops walking while the brain is working

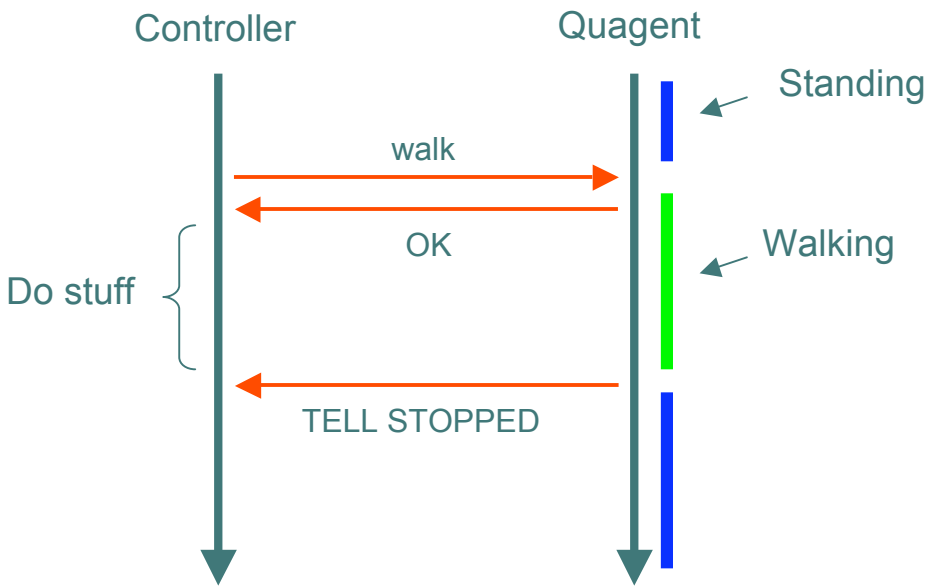
☞ neither of these alternatives is very desirable

☞ violates one of our central dogmas: be as realistic as possible

☞ we want both processes to be as unconstrained as possible so that each can perform their respective function as efficiently as possible

Interprocess Communication

Example: `...  q.walk(256);  ...`




```
...
Events events = null;
bool stopped = true;
...
q.walk(256);
stopped = getStopped(q.events());

while(!stopped) {
   // do stuff
   events = q.events();
   stopped = getStopped(events);
}
...
```

Event Polling

NOTE:
getStopped will return true if it finds the 'TELL STOPPED' event, otherwise it will return false.

IPC

```
class Asynch extends Quagent {

    static final int DIST = 20;

    public static void main(String[] args) throws Exception {
                new Asynch();
    }

    Asynch () throws Exception {
                super(); // run the constructer of the super class

                // action loop
                try {
                            while(true) {
                                        this.walk(5000);
                                        this.rays(1);
                                        handleEvents(this.events());
                                        Thread.currentThread().sleep(100);
                            }
                } catch (QDiedException e) { // the quagent died -- catch that exception
                            System.out.println("bot died!");
                }

                this.close();
    }
```

IPC

```
public void handleEvents(Events events) throws Exception {
        for (int ix = 0; ix < events.size(); ix++)
        {
                String e = events.eventAt(ix);

                if (e.indexOf("rays") >= 0)
                {
                        // NOTE: only works for single ray commands
                        // this is what the event looks like:
                        //   OK (ask rays 1) 1 worldspawn 379.969 54.342 0
                        // NOTE: parens are not included in tokens
                        String[] tokens = e.split("[()\\s]+");

                        double x = Double.parseDouble(tokens[6]);
                        double y = Double.parseDouble(tokens[7]);
                        double distance = Math.sqrt(x*x + y*y);

                        System.out.println("Distance: " + distance);

                        // if the distance is less than DIST ticks then turn 180 degrees
                        if (distance < DIST)
                                this.turn(180);
                }
        }
  }
}
```

Understanding the Problem Domain

Low

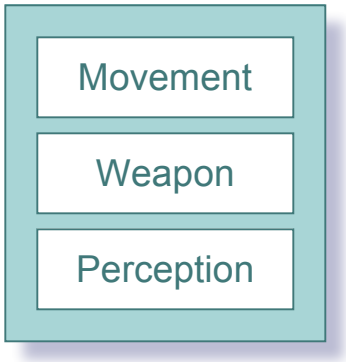
Abstraction Level

High

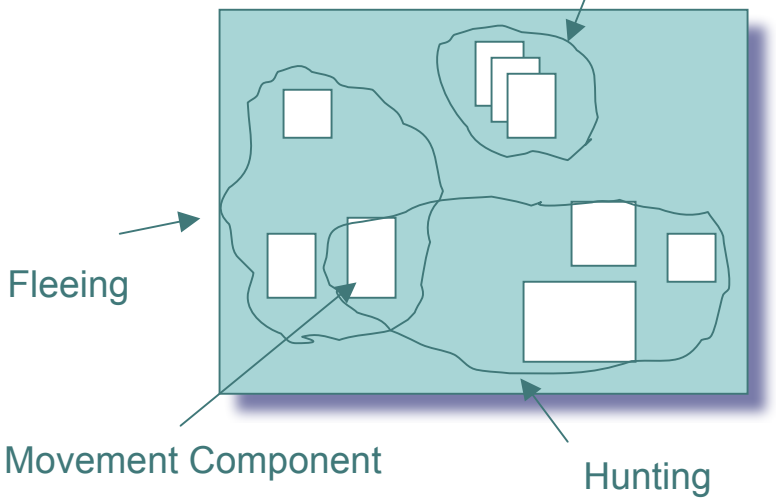
- Problem Decomposition
  - **Structural** – split problems according to the function of each component
  - **Behavioral** – split problems according to the set of activities required by the agent
  - **Goal** – split problems according to the goals envisioned for the agent

Understanding the Problem Domain

Structural Decomposition



Behavioral Decomposition



Goal decomposition	≡	Goals are possibly overlapping collections of behaviors.

Challenge Question

- Redesign your Warm-Up Lap algorithm to be as general as possible
  - Hint: use your knowledge of asynch IPC & problem decomposition
- Goal: from the spawn point move to a wall and walk one full lap along the wall.
- Challenges:
  - Rooms will be with angles of 90° and 270° with different sizes and shapes (i.e., not square)
  - Spawn points will be at different places in the rooms.