The table shows different versions of the interpretation loop, with the addition of step components, and related computational processes.

### What is Reflexion?

**Self-Awareness**

**Becoming** Aware of Something
(Temporally-Localized Event)

**Being Aware of Something**
(Temporally-Extended Process)

**Introspection Vs. Reflexion**

- Extreme (i.e., at every execution step) self-monitoring (i.e., periodic introspection)
- **virtual self-modification**

**Computational Self-Awareness**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Step Components</th>
<th>Process Creation</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>((S_L)_{-})</td>
<td>Lower Components</td>
<td>Process Creation</td>
<td>Process</td>
</tr>
<tr>
<td>((S_I, S_{LS}))</td>
<td>+ Single Introspection</td>
<td>Tracing</td>
<td>Execution Trace</td>
</tr>
<tr>
<td>((S_I, S_{LS}, S_{SU}))</td>
<td>+ Single Upper Step</td>
<td>Mirroring</td>
<td>Mirror Process</td>
</tr>
<tr>
<td>((S_I, D, S_{DU}))</td>
<td>(\rightarrow) Double Upper Step</td>
<td>Augmentation</td>
<td>Augmented Process</td>
</tr>
<tr>
<td>((S_I, D, S_{DU}))</td>
<td>(\rightarrow) Double Introspection</td>
<td>Reflexion</td>
<td>Reflexive Process</td>
</tr>
</tbody>
</table>

### Building up Computation Reflexion

**Higher-Order Theories of Consciousness:**

**Presence of the Self to Itself**

Damasio:

It is the process of an organism caught in the act of "representing its own changing state as it goes about representing something else."

Trautteur:

It is the process of an agent processing its own processing while processing an input.

**Philosophy of Mind**

**Computer Science**

**Self-Awareness**

**Being** Aware of Nothing

**Introspection** Vs. **Reflexion**

- Extremes (i.e., at every execution step) self-monitoring (i.e., periodic introspection)
- **virtual self-modification**

**Two key characteristics of computational reflexivity:**

- **Duality:** There are two concurrent processes (i.e., the target process and the augmented process).
- **Modularity:** It is a property of a particular type of interpreter, so it can be provided to any possible executable program.

**Prototype**

- We employed and modified the code of a Lisp meta-circular interpreter (the Lisp in Lisp). It is one of the simplest ways to implement a general-purpose interpreter.
- At **every stage of the computation**, the interpreter accesses and executes the code both **locally** (current function call) and **globally** (access to the main function code).
- The code of the target program can be modified at each step, thus influencing the next execution and performing the **virtual self-modification**.

The code of the proof-of-concept prototype is available at the URL: [http://valitutti.it/papers/reflexion/index.html](http://valitutti.it/papers/reflexion/index.html)