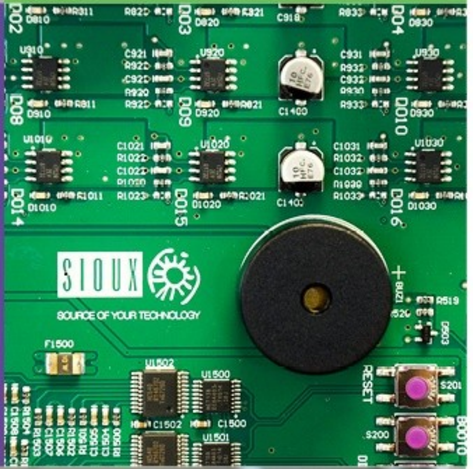
MECHATRONIC SOLUTIONS

MATHWARE

SIOUX

SOURCE OF YOUR TECHNOLOGY

ELECTRONIC SYSTEMS



TECHNICAL SOFTWARE

We bring **high-tech** to life

Invariant Dezyne Pattern

SOURCE OF YOUR TECHNOLOGY

Author: Joran Jessurun

20-05-2019

Invariant

```
interface Invariant {
    out void check();
    in void dummy();

    behaviour {
        on optional: check;
        on dummy: {}
    }
}

component InvariantDummy {
    provides Invariant dummy;
}
```

Example 1/2

```
interface Timer {
    extern ms $int$;
    in void create(ms duration);
    in void cancel();
    in bool isArmed();
    out void timeout();

    behaviour {
        bool armed = false;

        on isArmed: reply(armed);

        [!armed]
        {
            on create: armed = true;
            on cancel: {}
        }

        [armed]
        {
            on create: illegal;
            on cancel: armed = false;
            on inevitable: { timeout; armed = false; }
        }
    }
}
```

```
interface Monitor {
    in void start();
    in void stop();

    behaviour {
        bool running = false;

        [!running]
        {
            on start: running = true;
            on stop: {}
        }

        [running]
        {
            on start: illegal;
            on stop: running = false;
        }
    }
}
```

Example 2/2

```
component MonitorComponent {
    provides Monitor api;
    requires Timer timer;
    requires Invariant invariant;

    behaviour {
        bool running = false;

        void assert(bool check)
        {
            if(!check) illegal;
        }

        [!running] {
            on invariant.check(): {}

            on api.start(): { timer.create($1000$); running = true; }
            on api.stop(): {}
        }

        [running] {
            on invariant.check(): {
                bool armed = timer.isArmed();
                assert(armed);
            }

            on api.stop(): { timer.cancel(); running = false; }
            on timer.timeout(): { /* do stuff */ }
        }
    }
}
```

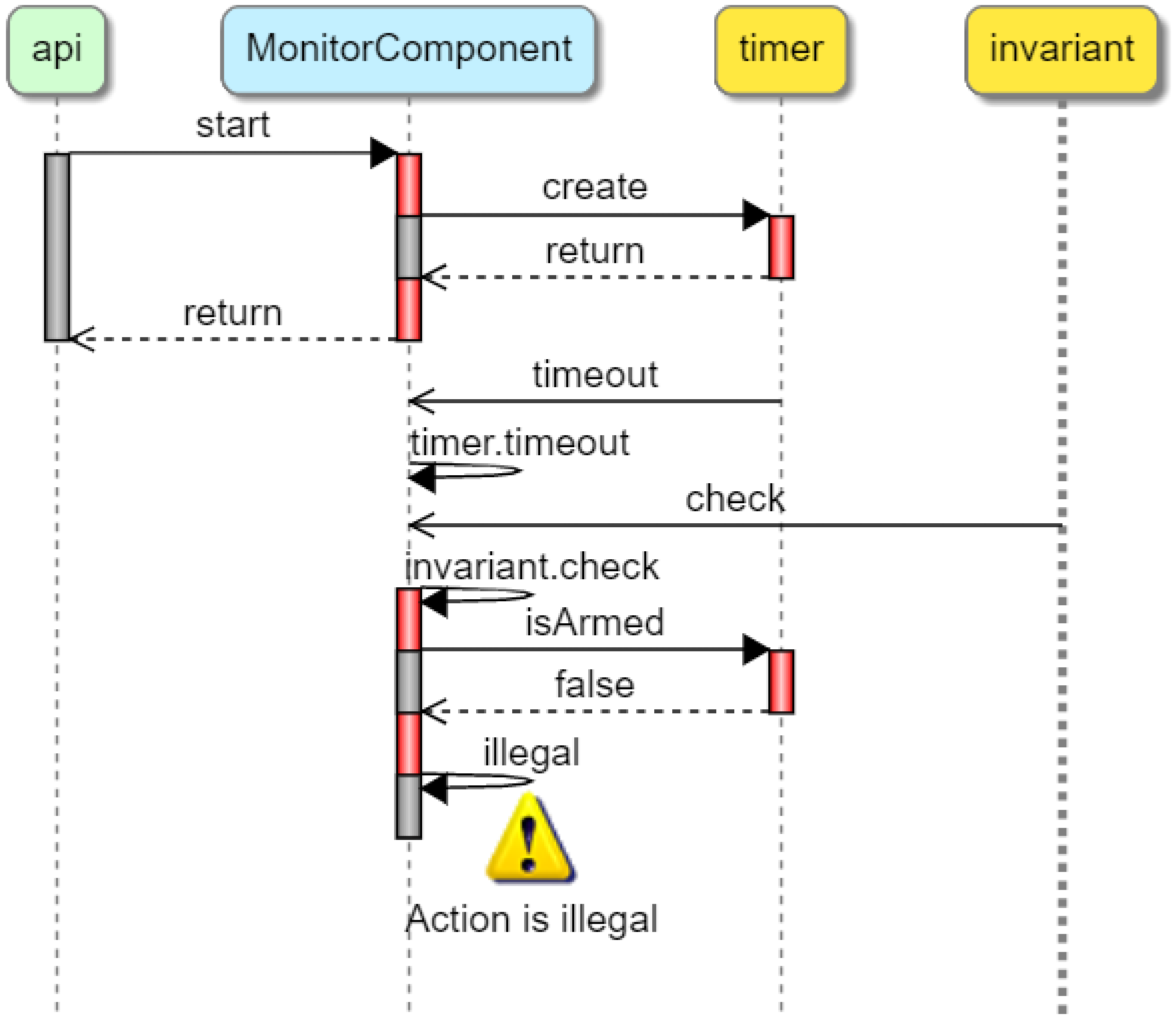
When the Monitor is running it should every second do something.

With the invariant we check that the timer is always active.

Example 2/2

```
component MonitorComponent {
    provides Monitor api;
    requires Timer timer;
    requires Invariant invariant;

    behaviour {
        bool running = false;

        void assert(bool check)
        {
            if(!check) illegal;
        }

        [!running] {
            on invariant.check(): {}

            on api.start(): { timer.create($1000$); running = tru
            on api.stop(): {}
        }

        [running] {
            on invariant.check(): {
                bool armed = timer.isArmed();
                assert(armed);
            }

            on api.stop(): { timer.cancel(); running = false; }
            on timer.timeout(): { /* do stuff */ }
        }
    }
}
```

`timer.create($1000$);`



QUESTIONS?

Source of your technology