Facilitating testing and monitoring of number entry systems in medical devices

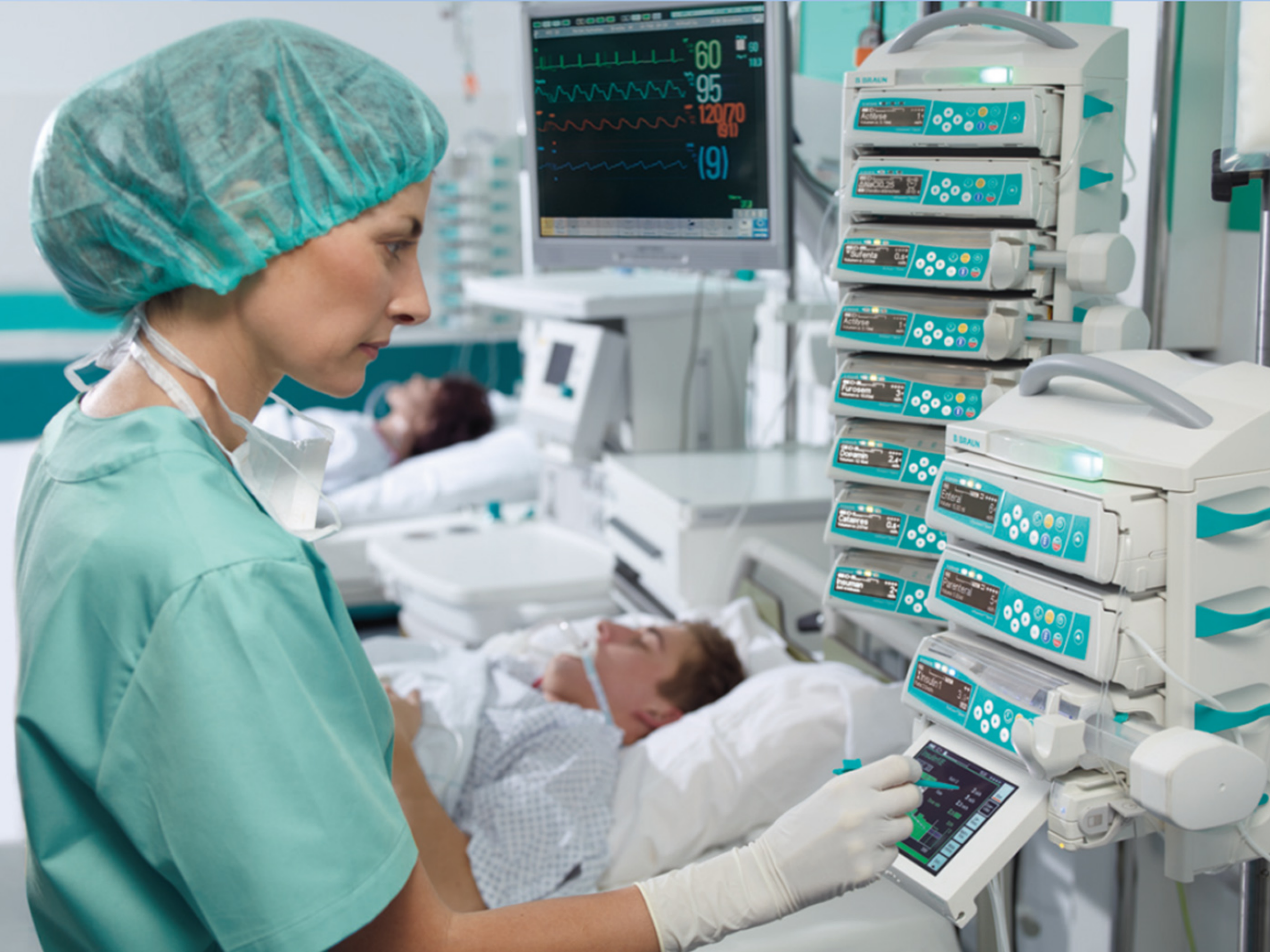
Abigail Cauchi, Christian Colombo,
Mark Micallef & Gordon Pace

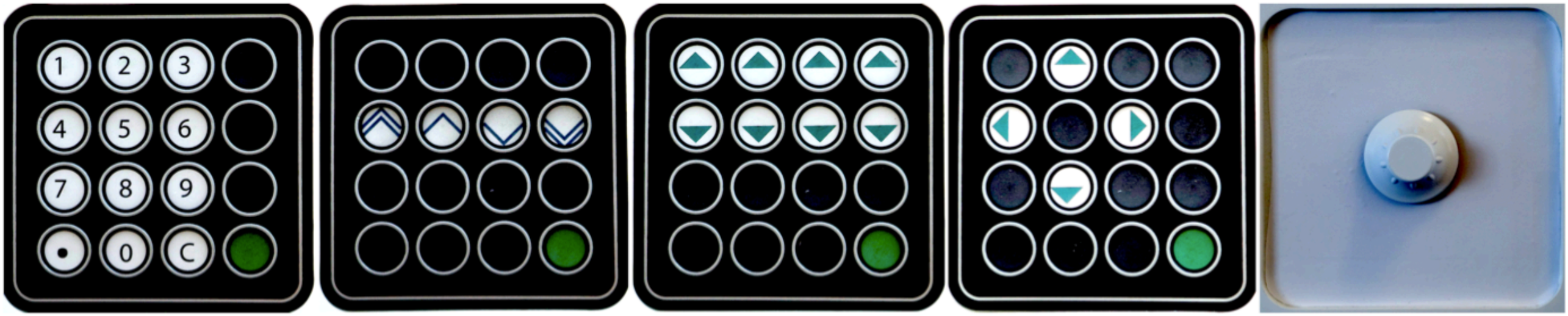
Xjenza The Malta Council for Science & Technology

Project GOMTA

ixaris

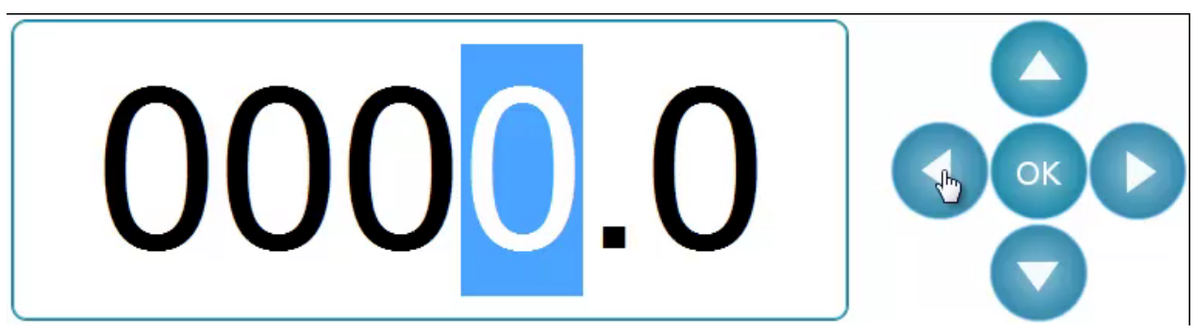
UNIVERSITY OF MALTA
L-Università ta' Malta

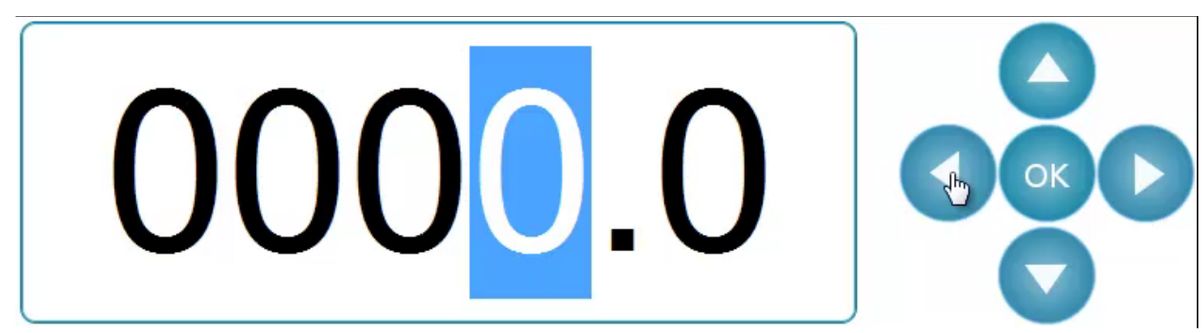

60
95
120/70
(9)
B.BRAUN
Actiflow
Furosem
Dopamin
Enteral
Parenteral
Insulin
B.BRAUN


1
2
3
4
5
6
7
8
9
0
C

Same input key sequence:

1. Left
2. Up 5 times
3. Left
4. Down



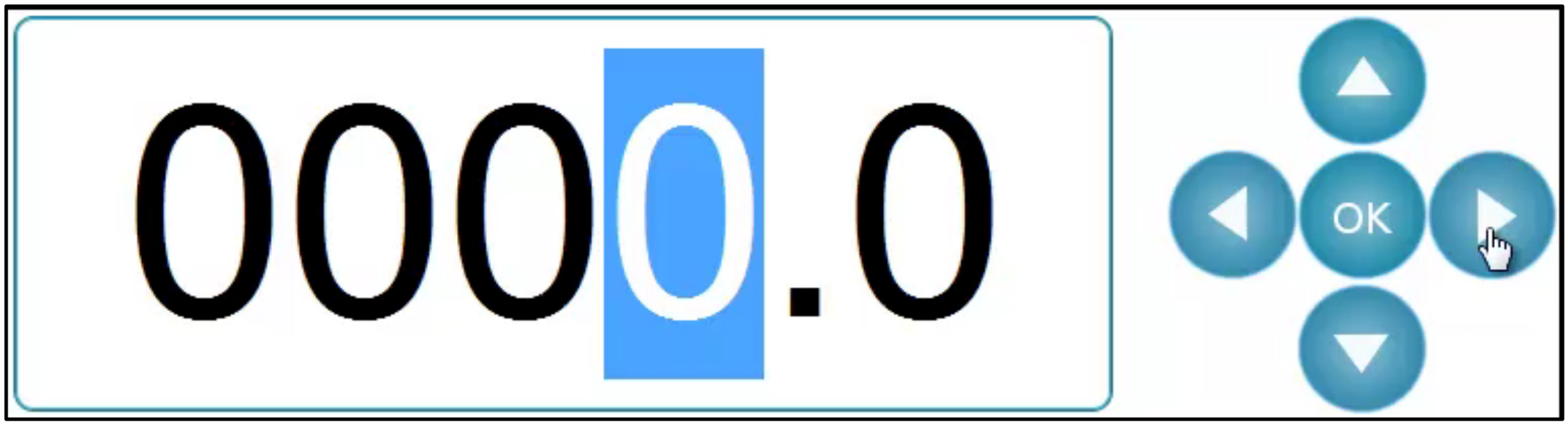


Under the UK Health & Safety At Work Act (1974) and under similar legislation in other countries, devices should be designed to reduce risk to be **As Low As Reasonably Practical**, ALARP.

UI principles

- No button presses with no effects
- Understandable error messages
- Reduce chances of (big) mistakes
- Etc

Cursor wraparound



The specification

- Coming from UI experts, psychologists, etc
- Implemented by software developers

The specification

- Coming from UI experts, psychologists, etc
- Implemented by software developers
- The problem is not new!

  Customers already specify systems

The specification

- Coming from UI experts, psychologists, etc
- Implemented by software developers

- The problem is not new!

Cus ns

Gherkin Feature

```
1  Feature: no digit or cursor wraparound but with error alerts
2
3    Scenario: Going beyond left boundary
4      Given cursor on leftmost position
5      When left is pressed
6      Then cursor position stays the same
7      And displayed number stays the same
8      And user is alerted
9
```

Automated Testing

- How can we use Gherkin to execute tests?

Gherkin output

```
@Given("^cursor on leftmost position$")
public void cursor_on_leftmost_position() throws Throwable {
    // Express the Regexp above with the code you wish you had
    throw new PendingException();
}

@When("^left is pressed$")
public void left_is_pressed() throws Throwable {
    // Express the Regexp above with the code you wish you had
    throw new PendingException();
}

@Then("^cursor position stays the same$")
public void cursor_position_stays_the_same() throws Throwable {
    // Express the Regexp above with the code you wish you had
    throw new PendingException();
}

@Then("^displayed number stays the same$")
public void displayed_number_stays_the_same() throws Throwable {
    // Express the Regexp above with the code you wish you had
    throw new PendingException();
}

@Then("^user is alerted$")
public void user_is_alerted() throws Throwable {
    // Express the Regexp above with the code you wish you had
    throw new PendingException();
}
```

StepDefinitions.java

```
3   public class StepDefinitions {
4
12      private fiveKey system = new fiveKey();
13
14      int cursorPos;
15      String displayedNumber;
16
17  @Given("^cursor on leftmost position$")
18  public void cursor_on_leftmost_position() throws Throwable {
19      system.setCursor(0);
20  }
21
22  @When("^left is pressed$")
23  public void left_is_pressed() throws Throwable {
24      this.cursorPos = system.getCursorPos();
25      this.displayedNumber = system.getDisplay();
26      system.left();
27  }
28
29  @Then("^cursor position stays the same$")
30  public void cursor_position_stays_the_same() throws Throwable {
31      Assert.assertTrue(this.cursorPos == system.getCursorPos());
32  }
33
34  @Then("^displayed number stays the same$")
35  public void displayed_number_stays_the_same() throws Throwable {
36      Assert.assertTrue(this.displayedNumber == system.getDisplay());
37  }
38
39  @Then("^user is alerted$")
40  public void user_is_alerted() throws Throwable {
41      Assert.assertTrue(system.isUserAlerted());
42  }
43
```

Automated Testing

- How can we use Gherkin to execute tests?

Execute the scenarios, invoking the steps

Logs from Running the Gherkin Feature

```
Feature: no digit or cursor wraparound but with error alerts

  Scenario: Going beyond left boundary  # C:/Users/User/Documents/NewEclipse Workspace/Cucumb
    Given cursor on leftmost position   # StepDefinitions.cursor_on_leftmost_position()
    When left is pressed                # StepDefinitions.left_is_pressed()
    Then cursor position stays the same # StepDefinitions.cursor_position_stays_the_same()
    And displayed number stays the same # StepDefinitions.displayed_number_stays_the_same()
    And user is alerted                 # StepDefinitions.user_is_alerted()

1 Scenarios (1 passed)
5 Steps (5 passed)
0m0.183s
```

Runtime Monitoring

Runtime Monitoring

- Can we use the tests to automatically generate monitors?

Gherkin Feature

```
1  Feature: no digit or cursor wraparound but with error alerts
2
3    Scenario: Going beyond left boundary
4      Given cursor on leftmost position
5      When left is pressed
6      Then cursor position stays the same
7      And displayed number stays the same
8      And user is alerted
9
```

Gherkin Feature

```
1  Feature: no digit or cursor wraparound but with error alerts
2
3    Scenario: Going beyond left boundary
4      Given cursor on leftmost position
5      When left is pressed
6      Then cursor position stays the same
7      And displayed number stays the same
8      And user is alerted
9
```

Precondition

Gherkin Feature

```
1  Feature: no digit or cursor wraparound but with error alerts
2
3    Scenario: Going beyond left boundary
4      Given cursor on leftmost position
5      When left is pressed
6      Then cursor position stays the same
7      And displayed number stays the same
8      And user is alerted
9
```

Precondition

Event

Gherkin Feature

```
1  Feature: no digit or cursor wraparound but with error alerts
2
3    Scenario: Going beyond left boundary
4      Given cursor on leftmost position
5      When left is pressed
6      Then cursor position stays the same
7      And displayed number stays the same
8      And user is alerted
9
```

Precondition

Event

Assertion

Aspect Oriented Programming Implementation

```
int around (fiveKey system): execution(* *.left(..)) && t
{
    if (system.getCursorPos() == 0){
        currentDisplay = system.getDisplay();
        cursorPos = 0;

        checkCursorLeft = true;
    }

    int ret = proceed(system);

    if(checkCursorLeft)
    {
        if ((system.getCursorPos() == cursorPos) && (system.isUserAlerted()) && (syst
            System.out.println("Monitor says: OK");
        else
            System.out.println("Monitor says: OOPS!");
    }
    return ret;

}
```

Event

Aspect Oriented Programming Implementation

```
int around (fiveKey system): execution(* *.left(..)) && t
{
    if (system.getCursorPos() == 0){
        currentDisplay = system.getDisplay();
        cursorPos = 0;

        checkCursorLeft = true;
    }

    int ret = proceed(system);

    if(checkCursorLeft)
    {
        if ((system.getCursorPos() == cursorPos) && (system.isUserAlerted()) && (syst
            System.out.println("Monitor says: OK");
        else
            System.out.println("Monitor says: OOPS!");
    }
    return ret;

}
```

Event

Precondition

Aspect Oriented Programming Implementation

```
int around (fiveKey system): execution(* *.left(..)) && t
{
    if (system.getCursorPos() == 0){
        currentDisplay = system.getDisplay();
        cursorPos = 0;

        checkCursorLeft = true;
    }

    int ret = proceed(system);

    if(checkCursorLeft)
    {
        if ((system.getCursorPos() == cursorPos) && (system.isUserAlerted()) && (syst
            System.out.println("Monitor says: OK");
        else
            System.out.println("Monitor says: OOPS!");
    }
    return ret;

}
```

Event

Precondition

Assertions

Aspect Oriented Programming Implementation

```
int around (fiveKey system): execu
{
    if (system.getCursorPos() =
        currentDisplay = sys
        cursorPos = 0;

        checkCursorLeft = true;
    }

    int ret = proceed(system);

    if(checkCursorLeft)
    {
        if ((system.getCursorPos() == cursorPos) && (system.isUserAlerted()) && (sys
            System.out.println("Monitor says: OK");
        else
            System.out.println("Monitor says: OOPS!");
    }
    return ret;

}
```

Crucial to match the precondition with the corresponding assertion

Assertions

Things to note

```
int around (fiveKey system): execution(* *.left(..))
{
    if (system.getCursorPos() == 0){
        currentDisplay = system.getDisplay();
        cursorPos = 0;

        checkCursorLeft = true;
    }

    int ret = proceed(system);

    if(checkCursorLeft)
    {
        if ((system.getCursorPos() == cursorPos) && (system.isUserAlerted()) && (syst
            System.out.println("Monitor says: OK");
        else
            System.out.println("Monitor says: OOPS!");
    }
    return ret;

}
```

Easy to extract

Precondition

Assertions

Things to not

```
int around (fiveKey system): execution(* *.l
{
    if (system.getCursorPos() == 0){
        currentDisplay = system.getDispl
        cursorPos = 0;

        checkCursorLeft = true;
    }

    int ret = proceed(system);

    if(checkCursorLeft)
    {
        if ((system.getCursorPos() == cursorPos) && (
            System.out.println("Monitor says: OK");
        else
            System.out.println("Monitor says: OOPS!");
    }
    return ret;

}
```

Not straight forward:

Going from action to condition

```
17 @Given("^cursor on leftmost position$")
18 public void cursor_on_leftmost_position(
19     system.setCursor(0);
20 }
21
```

One approach

```
12  boolean noCursorWrapAroundPreCondition = false;
13
14  after(int pos): (execution(* *.setCursor(..)) && args(pos)){
15      if (pos == 0)
16          noCursorWrapAroundPreCondition = true;
17      else
18          noCursorWrapAroundPreCondition = false;
19  }
20
```

One approach

```
12  boolean noCursorWrapAroundPreCondition = false;
13
14  after(int pos): (execution(* *.setCursor(..)) && args(pos)){
15      if (pos == 0)
16          noCursorWrapAroundPreCondition = true;
17      else
18          noCursorWrapAroundPreCondition = false;
19  }
20
```

```
27  int around (fiveKey system): execution(* *.left(..)) && target(system)
28  {
29      if (noCursorWrapAroundPreCondition){
30          currentDisplay = system.getDisplay();
31          cursorPos = 0;
32      }
33
34      int ret = proceed(system);
35
36      if(noCursorWrapAroundPreCondition)
37      {
38          if ((system.getCursorPos() == cursorPos) && (system.isUserAlerted()) && (
39              System.out.println("Monitor says: OK");
40          else
41              System.out.println("Monitor says: OOPS!");
42      }
43      return ret;
```

One approach

```
12  boolean noCursorWrapAroundPreCondition = false;
13
14  after(int pos): (execution(* *.setCursor(..)) && args(pos)){
15      if (pos == 0)
16          noCursorWrapAroundPreCondit      = true;
17      else
18          noCursorWrapAroundPreCond
19  }
20
```

```
27  int around (fiveKey sys
28  {
29      if (noCursorWrap
30          currentDispl
31          cursorPos = 0
32      }
33
34      int ret = proceed(system);
35
36      if(noCursorWrapAroundPreCondition)
37      {
38          if ((system.getCursorPos() == cursorPos) && (system.isUserAlerted()) && (
39              System.out.println("Monitor says: OK");
40          else
41              System.out.println("Monitor says: OOPS!");
42      }
43      return ret;
```

How does this interact with other methods?
Eg: Are there other methods which change th
e cursor position?
Does it affect other parts of the system state?

One approach

```
12  boolean noCursorWrapAroundPreCondition = false;
13
14  after(int pos): (execution(* *.setCursor(..)) && args(pos)){
15      if (pos == 0)
16          noCursorWrapAroundPreCondit      = true;
17      else
18          noCursorWrapAroundPreCond
19  }
20
```

```
27  int around (fiveKey sys
28  {
29      if (noCursorWra
30          currentDisp
31          cursorPos = 0
32      }
33
34      int ret = proceed(system);
35
36      if(noCursorWrapAroundPreCondition)
37      {
38          if ((system.getCursorPos() == cursorPos) && (system.isUserAlerted()) && (
39              System.out.println("Monitor says: OK");
40          else
41              System.out.println("Monitor says: OOPS!");
42      }
43      return ret;
```

Static analysis to identify such methods and reset the flag if detected

Left() method implementation

```
85      public  void left()
86      {
87          if (cursorPos > 0)
88          {
89              cursorPos --;
90          }
91          else{
92              isAlerted = true;
93          }
94
95      }
```

Aspect Oriented Programming Implementation

```
60      after(): (execution(* *.*(..)) && !cflow(adviceexecution())){
61          resetFlags();
62      }
63
64      public void resetFlags()
65      {
66          System.out.println("reseting flags");
67          checkCursorLeft = false;
68          checkCursorRight = false;
69      }
```

Discussion

- Resulting monitors can be too specific
- Can generalise using tester input but we want to automate

Options to consider

- Observing tests and extract invariants
  - Automatically deduce pre-post conditions:

    cursor == 0
    Left
    cursor == 0

Options to consider

- Observing tests and extract invariants
  - Automatically deduce pre-post conditions:

    cursor == 0
    Left
    cursor == 0

How much can/should you generalise from observing a few tests?

Filtering

- Filtering generated invari

vs

- Filtering resulting mo

- Attempting to model check the invariant to check if it is true
- Attempting to choose variables to consider for invariant extraction

Filtering

- Filtering generated invariants

vs

- Filtering resulting monitor alerts

- Showing the monitor results in priority order
- Asking user for feedback

Conclusions

- Numerous challenges when going from tests to monitors
- Right mix of techniques to obtain all available information: Static analysis, dynamic analysis on test + system description, test + system execution.
- Right mix of user input and automation