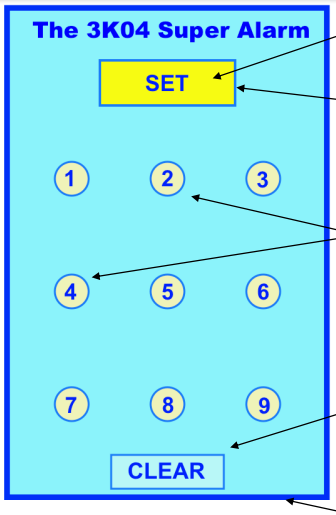
An Example

- We will now analyze a (scaled down) example of a home alarm system
- This example is meant to introduce you to the concepts and advantages of greater precision in requirements specification – and how we can build an implementation that we can demonstrate complies with its requirements

In Class Example



Light on when system is live

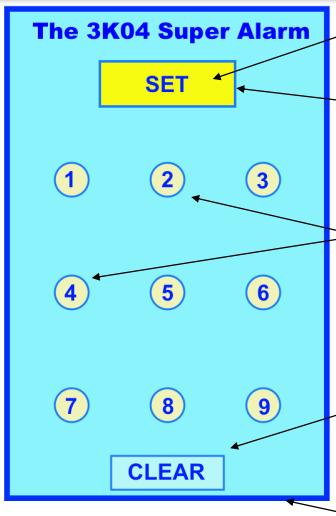
Button to make system live

Buttons to enter 3-digit code to stop alarm sounding & deactivate alarm

Button to clear erroneous digit entry
Have to re-enter all digits

Sensors connect to panel and alarm sounds when motion detected and system is live

In Class Example



Light on when system is live

Button to make system live

Let's document the requirements

Buttons to enter 3-digit code to stop alarm sounding & deactivate alarm

Button to clear erroneous digit entry
Have to re-enter all digits

Sensors connect to panel and alarm sounds when motion detected and system is live

72

72

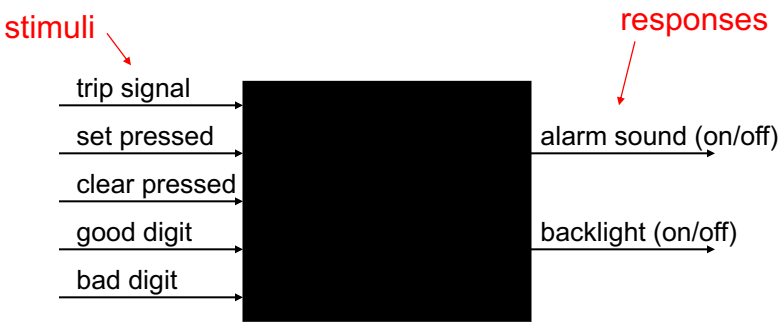
Example Requirements 1

1. The security system has a detector that sends a trip signal when motion is detected.
2. The security system is activated by pressing the SET button.
3. The SET button is illuminated when the system is active.
4. If a trip signal is detected while the system is active, an alarm is sounded.
5. A three-digit code must be entered to turn off the alarm sound.
6. Correct entry of the three-digit code deactivates the device.
7. If a mistake is made when entering the code, the user must press the CLEAR button before the entire code can be re-entered.

73

73

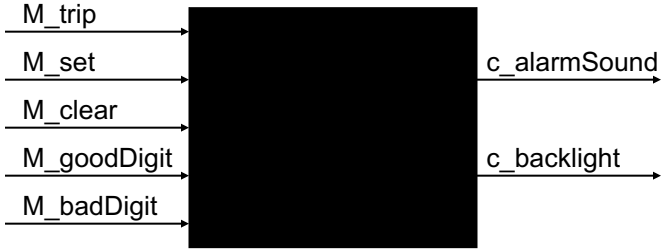
Notation



74

74

Notation



75

75

Input Sequence

- no stimulus
- M_set
- M_set . M_trip
- M_set . M_badDigit
- M_set . M_goodDigit
- M_set . M_trip . M_badDigit
- M_set . M_trip . M_goodDigit
- M_set . M_trip . M_goodDigit . M_goodDigit
- M_set . M_goodDigit . M_goodDigit

What is special about these sequences?

76

76

Input Sequence

- no stimulus
- M_set
- M_set . M_trip
- M_set . M_badDigit
- M_set . M_goodDigit
- M_set . M_trip . M_badDigit
- M_set . M_trip . M_goodDigit
- M_set . M_trip . M_goodDigit . M_goodDigit
- M_set . M_goodDigit . M_goodDigit

What is special about these sequences?

What if you append any of these legal stimuli to any of these sequences?

77

77

Input Sequence

- no stimulus
- M_set
- M_set . M_trip
- M_set . M_badDigit
- M_set . M_goodDigit
- M_set . M_trip . M_badDigit
- M_set . M_trip . M_goodDigit
- M_set . M_trip . M_goodDigit . M_goodDigit
- M_set . M_goodDigit . M_goodDigit

These are so-called "canonical" sequences. Append any (legal) stimulus to any of these sequences and you get one of these sequences as a result.

For example: M_set . M_badDigit . M_clear = M_set
(What about M_set . M_goodDigit . M_clear?

78

78

So how did that help us?

79

79

So how did that help us?

- We now assign response values for each one of these canonical sequences.
- Why does that help?

80

80

So how did that help us?

- We now assign response values for each one of these canonical sequences.
- Why does that help?
- Because now, for any sequence of stimuli, we know what the response should be.

81

81

Responses

	c_alarmSound	c_backlight
no stimulus	e_off	e_off
M_set	e_off	e_on
M_set . M_trip	e_on	e_on
M_set . M_badDigit	e_off	e_on
M_set . M_goodDigit	e_off	e_on
M_set . M_trip . M_badDigit	e_on	e_on
M_set . M_trip . M_goodDigit	e_on	e_on
M_set . M_trip . M_goodDigit . M_goodDigit	e_on	e_on
M_set . M_goodDigit . M_goodDigit	e_off	e_on

Is that now sufficient to specify the required behaviour?

82

82

Responses

	c_alarmSound	c_backlight
no stimulus	e_off	e_off
M_set	e_off	e_on
M_set . M_trip	e_on	e_on
M_set . M_badDigit	e_off	e_on
M_set . M_goodDigit	e_off	e_on
M_set . M_trip . M_badDigit	e_on	e_on
M_set . M_trip . M_goodDigit	e_on	e_on
M_set . M_trip . M_goodDigit . M_goodDigit	e_on	e_on
M_set . M_goodDigit . M_goodDigit	e_off	e_on

Is that now sufficient to specify the required behaviour?

No - we need to show effect of any new stimulus! How?

83

83

Responses

	c_alarmSound	c_backlight
no stimulus	e_off	e_off
M_set	e_off	e_on
M_set . M_trip	e_on	e_on
M_set . M_badDigit	e_off	e_on
M_set . M_goodDigit	e_off	e_on
M_set . M_trip . M_badDigit	e_on	e_on
M_set . M_trip . M_goodDigit	e_on	e_on
M_set . M_trip . M_goodDigit . M_goodDigit	e_on	e_on
M_set . M_goodDigit . M_goodDigit	e_off	e_on

Is that now sufficient to specify the required behaviour?

No - we need to show effect of any new stimulus! How?
One way: show current "state" & effect of each stimulus
(try it)

84

84

Effect of each stimulus

Notation:

no stimulus – initial condition	device off
M_set	device on
M_set . M_trip	alarm
M_set . M_badDigit	error
M_set . M_goodDigit	1 good
M_set . M_trip . M_badDigit	alarm & error
M_set . M_trip . M_goodDigit	alarm & 1 good
M_set . M_trip . M_goodDigit . M_goodDigit	alarm & 2 good
M_set . M_goodDigit . M_goodDigit	2 good

Behaviour (effect):

Current state

Current stimulus	device off {initial state}	device on	error	1 good	2 good	alarm	alarm & error	alarm & 1 good	alarm & 2 good
M_badDigit	-	error	-	error	error	alarm & error	-	alarm & error	alarm & error
M_clear	-	-	device on	device on	device on	-	alarm	alarm	alarm
M_goodDigit	-	1 good	-	2 good	device off	alarm & 1 good	-	alarm & 2 good	device off
M_set	device on	-	-	-	-	-	-	-	-
M_trip	-	alarm	alarm & error	alarm	alarm	-	-	-	-

85

85

Questions

- Why did we use M_, c_, e_ etc in our notation?
- Does it make sense to have M_badDigit and M_goodDigit as our stimuli?
- How does the sensor value that detects motion get into our software?
- How about the “secret” code to switch off the device or alarm sound? How is it conveyed to the device?

86

86

Notation

- We already saw Parnas and Madey’s 4 variable model that describes how requirements are related to the software design. We call stimuli “monitored variables”, and responses “controlled variables”. Our notation simply uses prefixes to describe the “kind” of identifier, hence M_ or m_ for monitored variables (M_ for time discrete, m_ for time continuous), C_ or c_ for controlled variables, etc. e_ is used for enumerated tokens (think “enum” in C, C# etc)

87

87

M_badDigit & M_goodDigit

- It does not make sense to use M_badDigit and M_goodDigit as monitored variables. Why not?

88

88

M_badDigit & M_goodDigit

- It does not make sense to use M_badDigit and M_goodDigit as monitored variables. Why not?
  - They are not really monitored variables
  - The true monitored variables are the digits ‘1’, ‘5’ etc
  - We used M_badDigit and M_goodDigit because it simplified our description
  - We should now determine how to “derive” badDigit and goodDigit given a monitored variable M_digit for instance, where M_digit is one of ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’, ‘0’

89

89

Determining badDigit & goodDigit - 1

- Given that the system knows the secret code (assume a string of 3 characters called secretCode):

		Result
No digit since initialization or since M_clear		No Change
One digit since initialization or since M_clear	c == secretCode[0]	goodDigit
	c != secretCode[0]	badDigit
Two digits since initialization or since M_clear	c == secretCode[1]	goodDigit
	c != secretCode[1]	badDigit
Three digits since initialization or since M_clear	c == secretCode[2]	goodDigit
	c != secretCode[2]	badDigit
>Three digits since initialization or since M_clear		No Change

which leads us to …

90

90

Determining badDigit & goodDigit - 2

- So, for current state, given c from M_digit:

		Result
device off || error || alarm & error		No Change
device on || alarm	c == secretCode[0]	goodDigit
	c != secretCode[0]	badDigit
1 good || alarm & 1 good	c == secretCode[1]	goodDigit
	c != secretCode[1]	badDigit
2 good || alarm & 2 good	c == secretCode[2]	goodDigit
	c != secretCode[2]	badDigit

- One way to do this is to make a different monitored variable for each digit, eg
  M_one == > c = '1', etc

91

91

M_trip

- In a real device, there would be hardware that detects motion and then an interface to the software that delivers the value (in this case just the equivalent of a boolean) so that the software can work with it
- In our case, we will just make a button that the user can click to trigger M_trip

92

92

The Secret Code

- There are many ways of entering the secret code
- To start, let us modify the requirements just a little, to say that a string of 3 characters (numeric only) must be entered when the Set button is pressed. Therefore, pressing the Set button will result in entering the secret code and then switching the device on
- In a later version we will modify the requirements to make the behaviour more realistic – the Set button will have 2 modes: Set and Code. Set will start the device. Code will let the user enter the secret code by pressing the digit buttons

93

93

Anything Else?

- Have we coped with everything?

94

94

Anything Else?

- Have we coped with everything?
- No. We need some way of showing that the alarm is sounding
- At this stage, we will simply have a red area on the form to indicate the alarm is sounding
- Later we could add sound …

95

95

Using digits to enter secret code

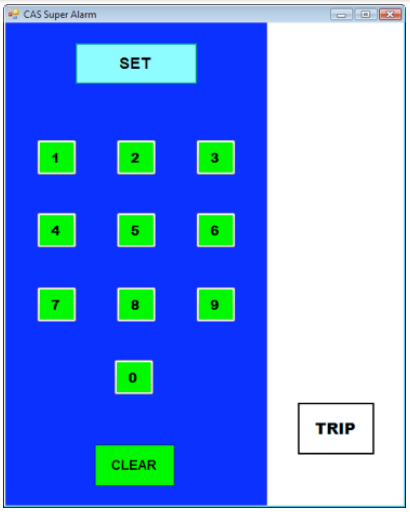
- M_mode . M_digit(c0) . M_digit(c1) . M_digit(c2) ➔ M_set, secretCode = c0+c1+c2
-
- Replace M_set by M_mode in transition table

96

96

CAS Super Alarm in Visual Studio - 1

Initial state

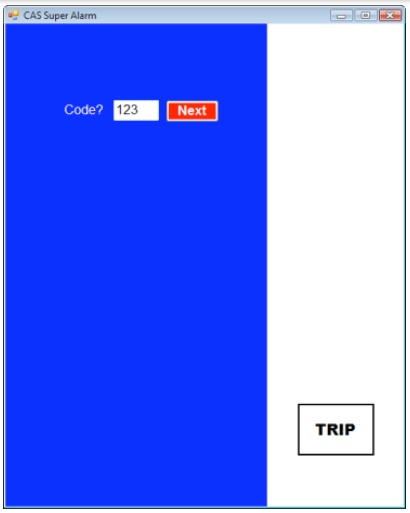


97

97

CAS Super Alarm in Visual Studio - 2

Entering the secret code

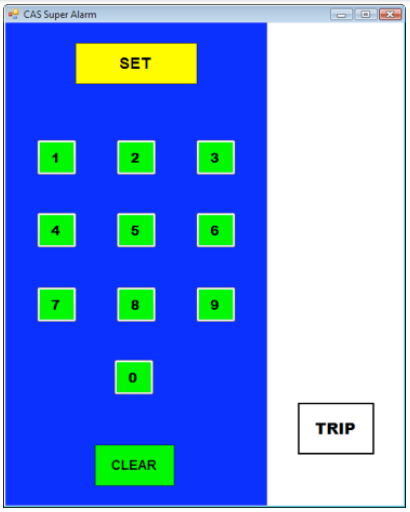


98

98

CAS Super Alarm in Visual Studio - 3

Device active

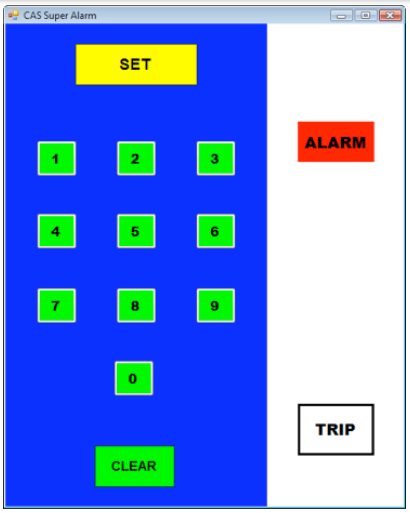


99

99

CAS Super Alarm in Visual Studio - 4

Alarm sounding after Trip was pressed



100

100

CAS Super Alarm in Visual Studio - 5

- First attempt
  - Use M_goodDigit and M_badDigit as inputs to the class that you create to implement the CAS alarm.
  - Use a textbox on the form to input the secret code – shown in the next slide. After user enters the code, pressing "NEXT" will check that the code is legal. If yes it starts the alarm device. If not, clears textbox and user tries again. (Display an appropriate error message.)

101

101

CAS Super Alarm in Visual Studio - 6

- Now try the same thing but without making a textbox for the secret code.
- This time:
  - User presses “SET”, enters a mode where we can set the secret code.
  - Button label changes to “CODE”.
  - Next 3 digits pressed define the secret code.
  - After the 3rd one is pressed, button changes to “SET” and is backlit in yellow.
- Also, use M_one, M_two etc as the variables input to the CAS alarm class.

102