Design a Quartz Watch using Atom3 DCharts Formalism

WeiBin Liang
School of Computer Science
McGill University
April 4, 2004

Overview

- Atom3 DCharts Formalism
- Statecharts virtual machine (SVM) and Statecharts compiler compiler (SCC)
- A design of a simplified quartz watch
- Demo

Atom3 DCharts

- A subset of StateCharts formalism supported by Atom3:
  - David Harel's State charts
  - Transition priority
  - Model importation
  - Macros

SVM and SCC

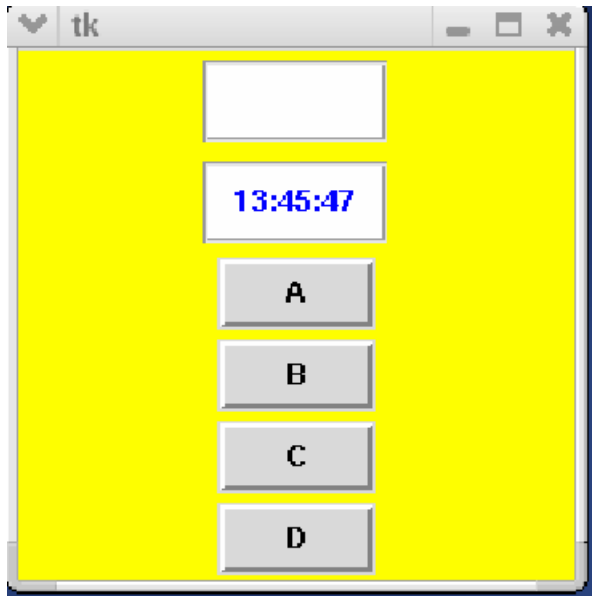
- SVM simulates a DCharts model in real time
  - Initializes the model
  - Simulates the model by running its interactor
  - Finalizes the model
- SCC parses the model description file and automatically synthesis code in target languages

Example: A Quartz Watch

- Requirements: (It's an simplified version of David Harel's Citizen quartz watch)
  - Time
  - Alarm
- Steps:
  - UI design
  - Model the control component
  - Bridge the gap between the static and dynamic components

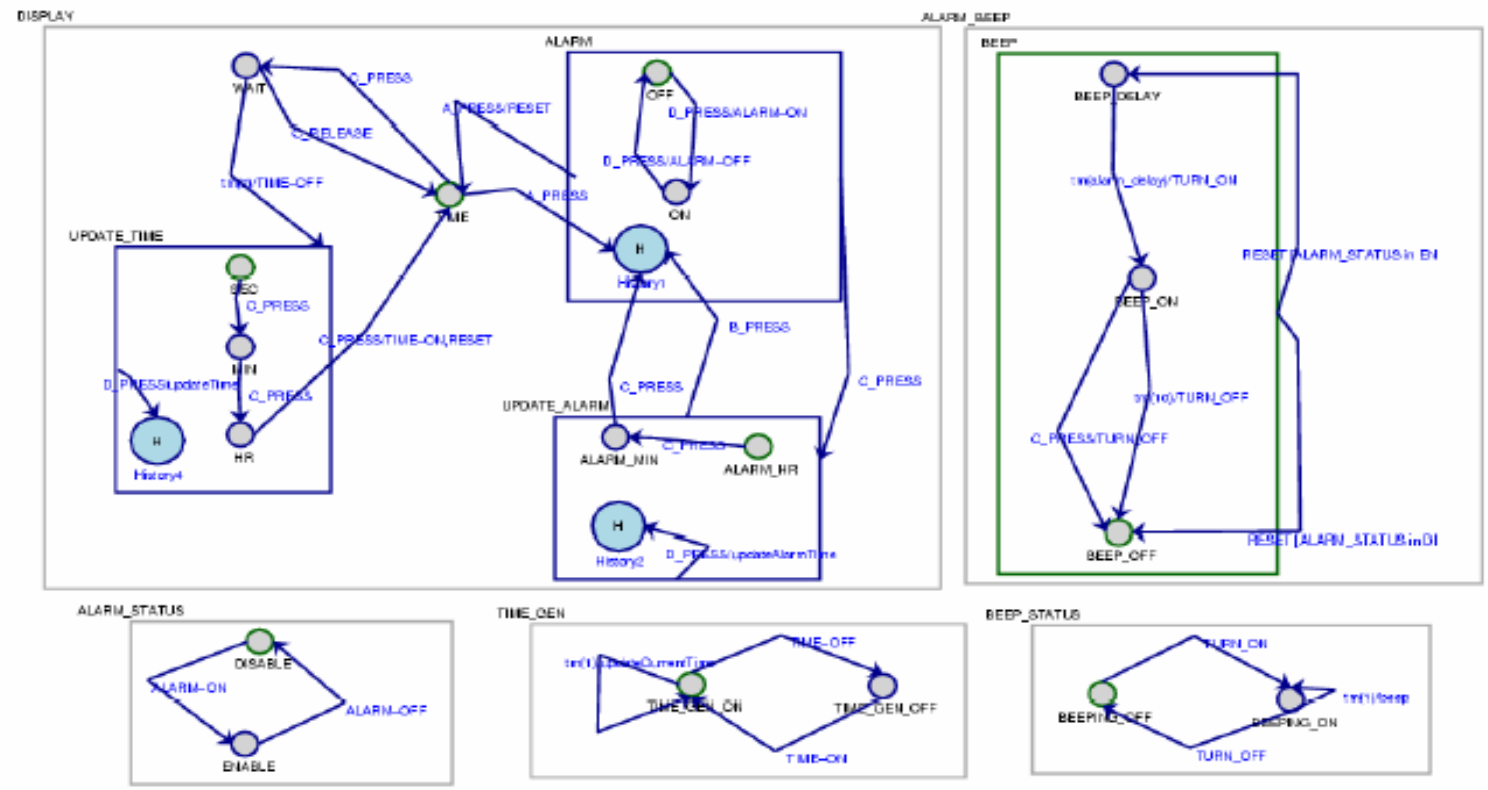
Example: A Quartz Watch (2)

- UI



Example: A Quartz Watch (3)

- Model the dynamic behavior of the system



Example: A Quartz Watch (4)

- Design issues:
  - Models the current time
  - Models the alarm time
- GUI based design Vs. text based design

Example: A Quartz Watch (5)

- Bridge the gap
  - Alternative 1:
    - The gui holds a reference to the behavior object and vice versa
  - Alternative 2:
    - Design a Proxy class that holds private attributes gui and behavior, and only those necessary methods will be exposed

Sample Code: Proxy.py

```
from Time import *

class Proxy:
    def __init__(self):
        self.__gui = None
        self.__behavior = None

    def setGUI(self, gui):
        self.__gui = gui

    def setBehavior(self, behavior):
        self.__behavior = behavior

def event(self, eventName):
        self.__behavior.event(eventName)

def displayTime(self, t):
        self.__gui.displayTime(str(t))

def displayAlarmStatus(self, state):
        self.__gui.displayAlarmStatus(state)

def beep(self):
    print '\a'
    self.__gui.blink()
```

Demo

Future work

- What we want from an extension of Atom3 DCharts
  - Importation of helper methods for action code
  - Ways to define attributes that can be used by the state machine model

Reference

[1] David Harel, StateCharts: A visual formalism for complex system. Science of Computer Programming 8 (1987) 231-274.

[2] Thomas Huining Feng: SVM and SCC Tutorial.

http://moncs.cs.mcgill.ca/people/tfeng/