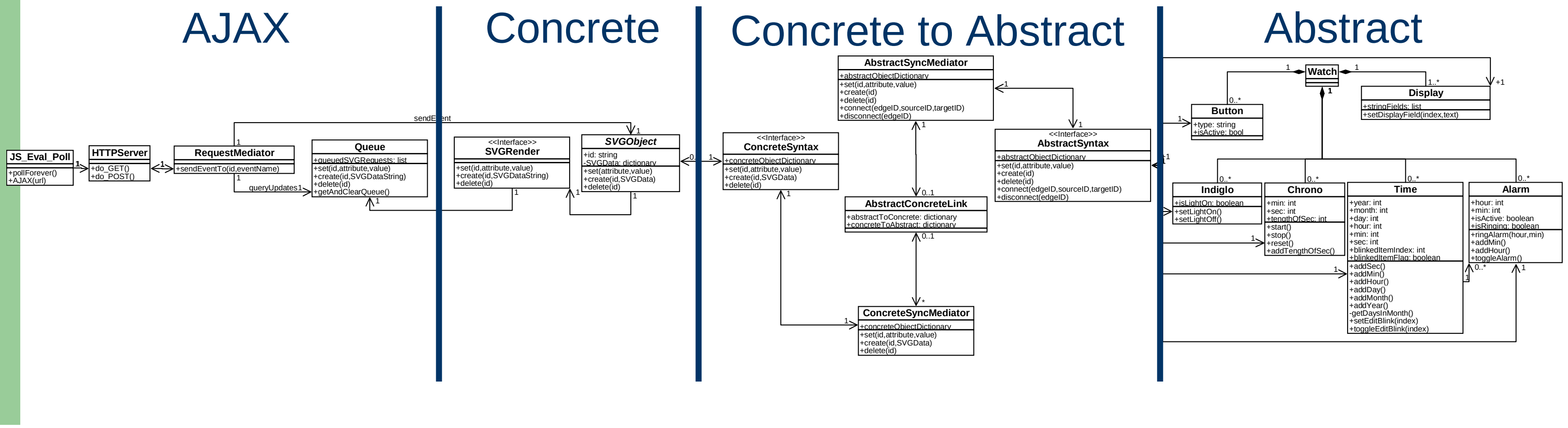
Modelling and Synthesizing Visual Modelling Environments

Denis Dubé

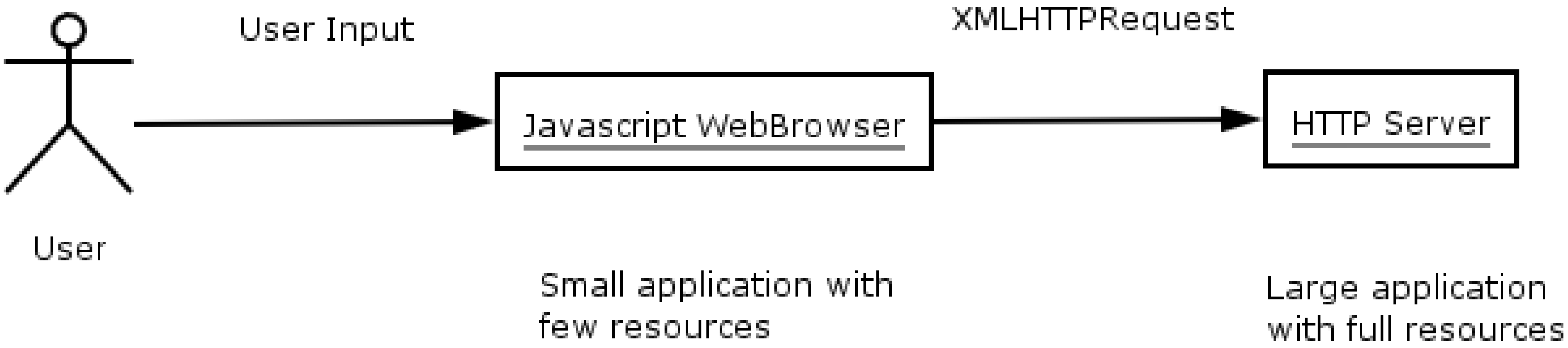
Outline

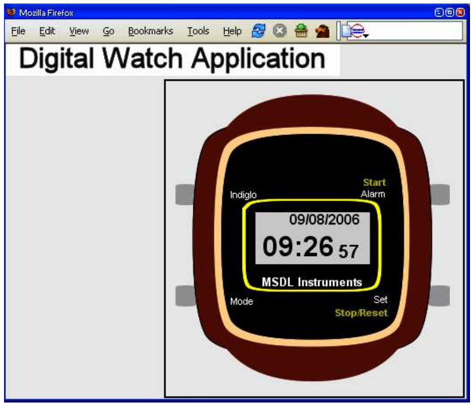
- **AJAX Introduction**
- Abstract Models to AJAX
    - Synthesizing a Digital Watch Application

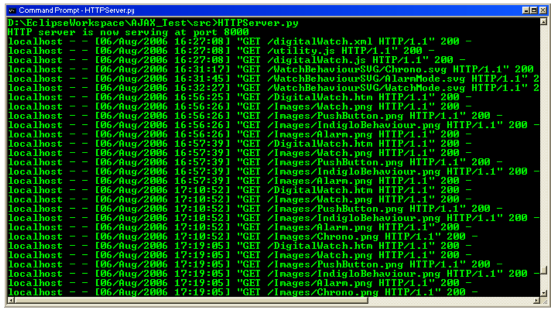


Introducing AJAX

- Asynchronous JavaScript and XML


User Input
XMLHTTPRequest
User
Javascript WebBrowser
HTTP Server
Small application with few resources
Large application with full resources





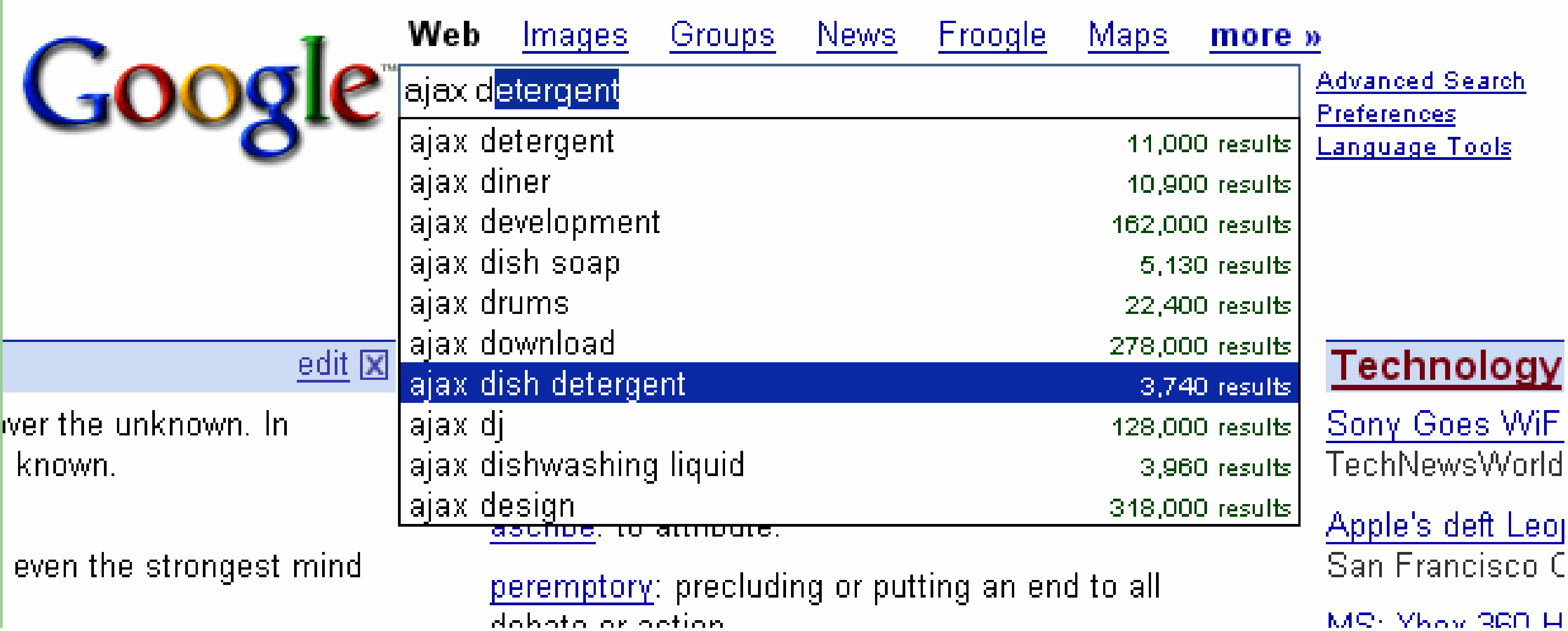
```
D:\EclipseWorkspace\AJAX_Test\src>HTTPServer.py
HTTP server is now serving at port 8000
localhost - - [06/Aug/2006 16:27:08] "GET /digitalWatch.xml HTTP/1.1" 200 -
localhost - - [06/Aug/2006 16:27:08] "GET /utility.js HTTP/1.1" 200 -
localhost - - [06/Aug/2006 16:27:08] "GET /digitalWatch.js HTTP/1.1" 200 -
localhost - - [06/Aug/2006 16:31:17] "GET /WatchBehaviourSVG/Chrono.svg HTTP/1.1" 200
localhost - - [06/Aug/2006 16:31:45] "GET /WatchBehaviourSVG/AlarmMode.svg HTTP/1.1" 2
localhost - - [06/Aug/2006 16:32:27] "GET /WatchBehaviourSVG/WatchMode.svg HTTP/1.1" 2
localhost - - [06/Aug/2006 16:56:25] "GET /DigitalWatch.htm HTTP/1.1" 200 -
localhost - - [06/Aug/2006 16:56:26] "GET /Images/Watch.png HTTP/1.1" 200 -
localhost - - [06/Aug/2006 16:56:26] "GET /Images/PushButton.png HTTP/1.1" 200 -
localhost - - [06/Aug/2006 16:56:26] "GET /Images/IndigloBehaviour.png HTTP/1.1" 200 -
localhost - - [06/Aug/2006 16:56:26] "GET /Images/Alarm.png HTTP/1.1" 200 -
localhost - - [06/Aug/2006 16:57:39] "GET /DigitalWatch.htm HTTP/1.1" 200 -
localhost - - [06/Aug/2006 16:57:39] "GET /Images/Watch.png HTTP/1.1" 200 -
localhost - - [06/Aug/2006 16:57:39] "GET /Images/PushButton.png HTTP/1.1" 200 -
localhost - - [06/Aug/2006 16:57:39] "GET /Images/IndigloBehaviour.png HTTP/1.1" 200 -
localhost - - [06/Aug/2006 16:57:39] "GET /Images/Alarm.png HTTP/1.1" 200 -
localhost - - [06/Aug/2006 17:10:52] "GET /DigitalWatch.htm HTTP/1.1" 200 -
localhost - - [06/Aug/2006 17:10:52] "GET /Images/Watch.png HTTP/1.1" 200 -
localhost - - [06/Aug/2006 17:10:52] "GET /Images/PushButton.png HTTP/1.1" 200 -
localhost - - [06/Aug/2006 17:10:52] "GET /Images/IndigloBehaviour.png HTTP/1.1" 200 -
localhost - - [06/Aug/2006 17:10:52] "GET /Images/Alarm.png HTTP/1.1" 200 -
localhost - - [06/Aug/2006 17:10:52] "GET /Images/Chrono.png HTTP/1.1" 200 -
localhost - - [06/Aug/2006 17:19:05] "GET /DigitalWatch.htm HTTP/1.1" 200 -
localhost - - [06/Aug/2006 17:19:05] "GET /Images/Watch.png HTTP/1.1" 200 -
localhost - - [06/Aug/2006 17:19:05] "GET /Images/PushButton.png HTTP/1.1" 200 -
localhost - - [06/Aug/2006 17:19:05] "GET /Images/IndigloBehaviour.png HTTP/1.1" 200 -
localhost - - [06/Aug/2006 17:19:05] "GET /Images/Alarm.png HTTP/1.1" 200 -
localhost - - [06/Aug/2006 17:19:05] "GET /Images/Chrono.png HTTP/1.1" 200 -
```

JavaScript Can:

- Add/Remove/Modify any element of the XML structure of a web page
  - Essentially anything you can do in a web page development tool, JavaScript can do on the fly
- Capture user input
- Send XMLHTTPRequests a la AJAX

AJAX Simple Example

- Google Suggest: Attempts to complete search query as user types in each character



ajax detergent

ajax detergent	11,000 results
ajax diner	10,900 results
ajax development	162,000 results
ajax dish soap	5,130 results
ajax drums	22,400 results
ajax download	278,000 results
ajax dish detergent	3,740 results
ajax dj	128,000 results
ajax dishwashing liquid	3,960 results
ajax design	318,000 results

wer the unknown. In known.

even the strongest mind

ascribe: to attribute.

peremptory: precluding or putting an end to all debate or action

Technology

Sony Goes WiF

TechNewsWorld

Apple's deft Leo

San Francisco C

MS: Xbox 360 H

JavaScript (the J in AJAX)

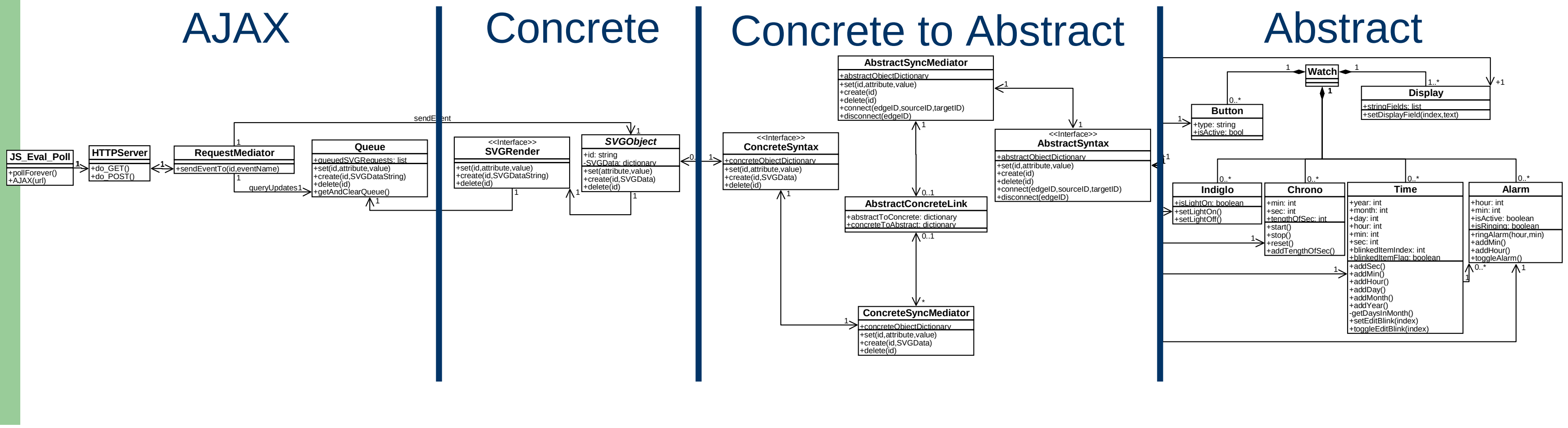
- Good:
  - WebBrowsers (Firefox, Netscape, Opera, IE, etc.) all include JavaScript interpreters
- Bad:
  - Not all clients have JavaScript enabled
  - JavaScript is SLOW → Three orders of magnitude slower than compiled C code
  - Suffers from periodic slowdowns (on Firefox at least)
- NOTE: JavaScript and Java have very little in common (save their names)

JavaScript Can Not:

- Open/Save/Execute files on the client
- Open a pipe, TCP/IP, or any kind of networking connection on the client
- Fully support object oriented programming
- Beep the PC speaker ☹

Outline

- AJAX Introduction
- **Abstract Models to AJAX**
  - **Synthesizing a Digital Watch Application**



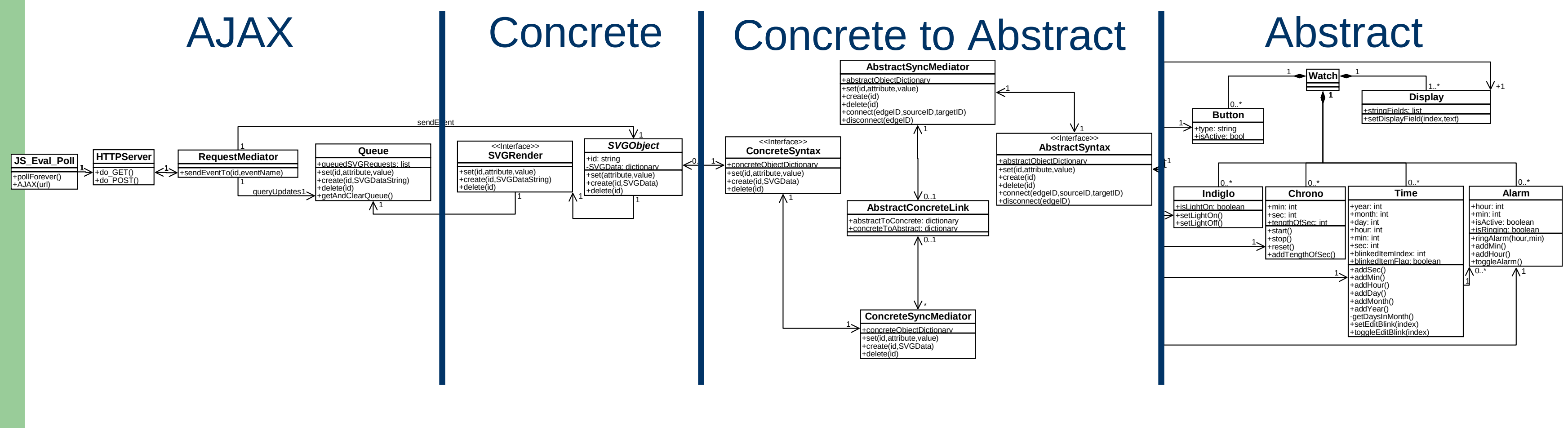
Synthesizing a Digital Watch

- AJAX has enormous potential

- Existing AJAX frameworks do not currently leverage the power of modelling (i.e.: coding by hand is the norm)

Overview

- The modelling process for arbitrarily complex AJAX applications (including AToM^3) can be shown with a seemingly simple Digital Watch example with four major components:



AJAX

JS_Eval_Poll
+pollForever()
+AJAX(url)

HTTPServer
+do_GET()
+do_POST()

RequestMediator
+sendEventTo(id,eventName)

Queue
+queuedSVGRequests: list
+set(id,attribute,value)
+create(id,SVGDataString)
+delete(id)
+getAndClearQueue()

sendEvent

queryUpdates

Concrete

<<Interface>>
SVGRender
+set(id,attribute,value)
+create(id,SVGDataString)
+delete(id)

SVGObject
+id: string
-SVGData: dictionary
+set(attribute,value)
+create(id,SVGData)
+delete(id)

Concrete to Abstract

<<Interface>>
ConcreteSyntax
+concreteObjectDictionary
+set(id,attribute,value)
+create(id,SVGData)
+delete(id)

AbstractSyncMediator
+abstractObjectDictionary
+set(id,attribute,value)
+create(id)
+delete(id)
+connect(edgeID,sourceID,targetID)
+disconnect(edgeID)

AbstractConcreteLink
+abstractToConcrete: dictionary
+concreteToAbstract: dictionary

ConcreteSyncMediator
+concreteObjectDictionary
+set(id,attribute,value)
+create(id,SVGData)
+delete(id)

<<Interface>>
AbstractSyntax
+abstractObjectDictionary
+set(id,attribute,value)
+create(id)
+delete(id)
+connect(edgeID,sourceID,targetID)
+disconnect(edgeID)

Abstract

Watch

Button
+type: string
+isActive: bool

Display
+stringFields: list
+setDisplayField(index,text)

Indiglo
+isLightOn: boolean
+setLightOn()
+setLightOff()

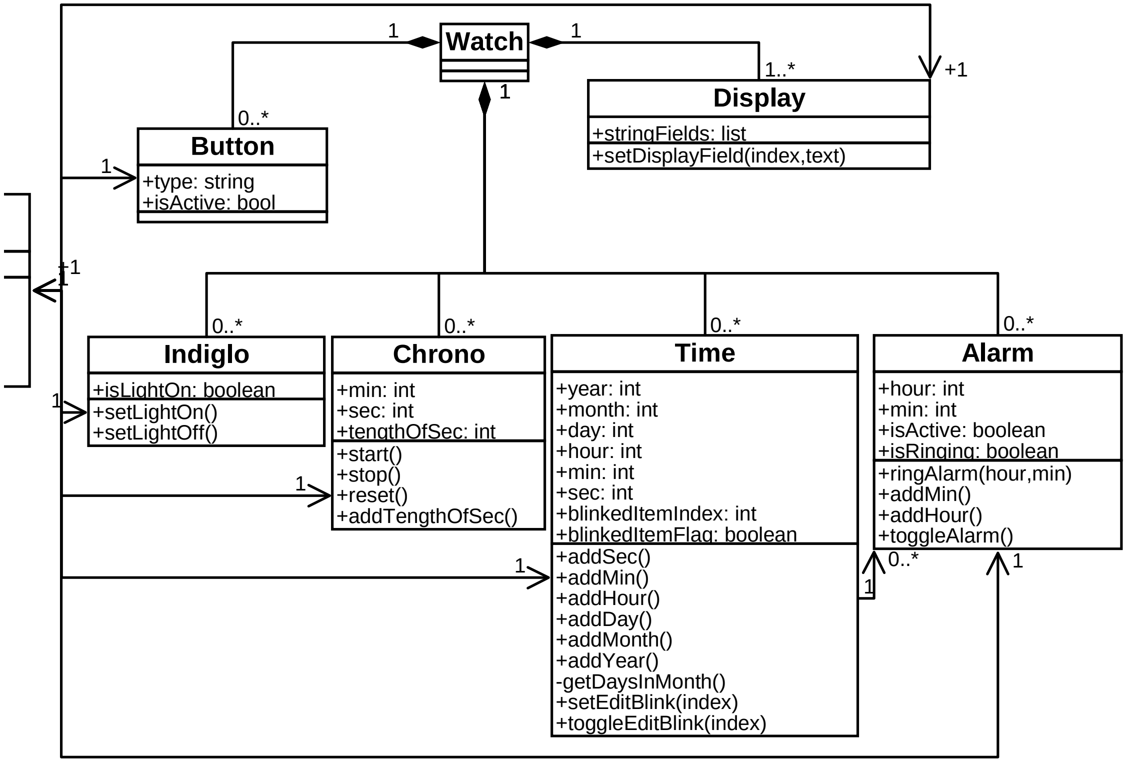
Chrono
+min: int
+sec: int
+tengthOfSec: int
+start()
+stop()
+reset()
+addTengthOfSec()

Time
+year: int
+month: int
+day: int
+hour: int
+min: int
+sec: int
+blinkedItemIndex: int
+blinkedItemFlag: boolean
+addSec()
+addMin()
+addHour()
+addDay()
+addMonth()
+addYear()
-getDaysInMonth()
+setEditBlink(index)
+toggleEditBlink(index)

Alarm
+hour: int
+min: int
+isActive: boolean
+isRinging: boolean
+ringAlarm(hour,min)
+addMin()
+addHour()
+toggleAlarm()

Abstract Syntax 1

- Modelled using UML Class Diagrams



Watch

Button
+type: string
+isActive: bool

Display
+stringFields: list
+setDisplayField(index,text)

Indiglo
+isLightOn: boolean
+setLightOn()
+setLightOff()

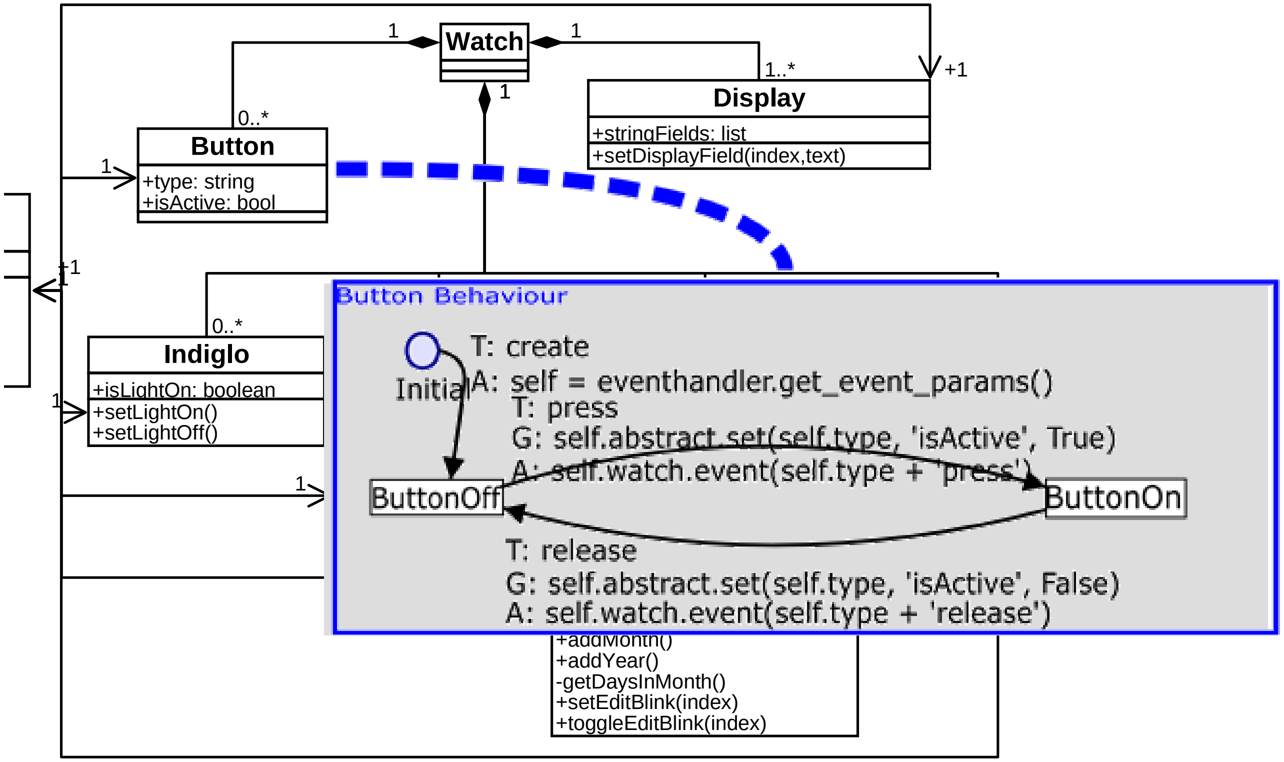
Chrono
+min: int
+sec: int
+tengthOfSec: int
+start()
+stop()
+reset()
+addTengthOfSec()

Time
+year: int
+month: int
+day: int
+hour: int
+min: int
+sec: int
+blinkedItemIndex: int
+blinkedItemFlag: boolean
+addSec()
+addMin()
+addHour()
+addDay()
+addMonth()
+addYear()
-getDaysInMonth()
+setEditBlink(index)
+toggleEditBlink(index)

Alarm
+hour: int
+min: int
+isActive: boolean
+isRinging: boolean
+ringAlarm(hour,min)
+addMin()
+addHour()
+toggleAlarm()

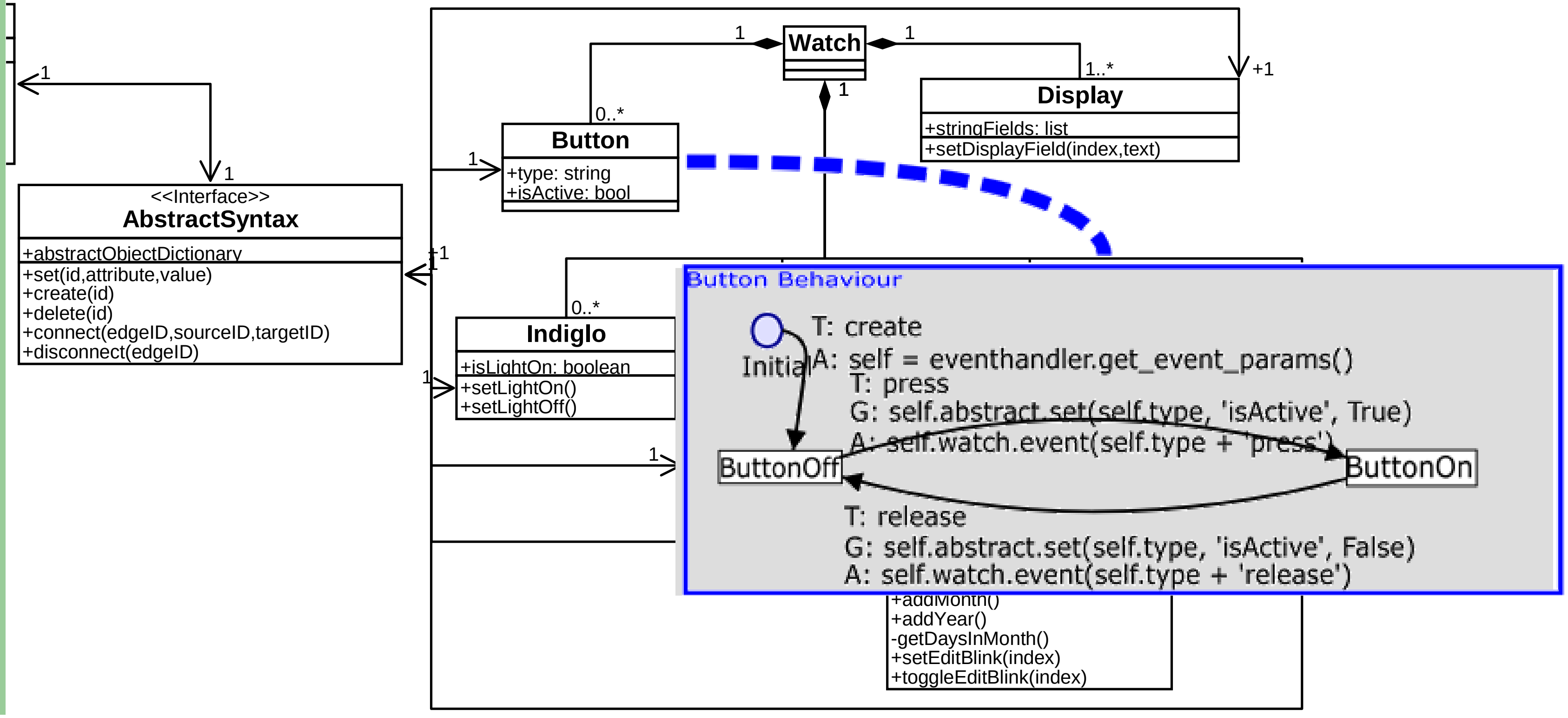
Abstract Syntax 2

- UML Class Diagrams and State Charts



Abstract Syntax 3

- Abstract attribute changes are forwarded to an interface for model checking and visual updates



<<Interface>>
AbstractSyntax
+abstractObjectDictionary
+set(id,attribute,value)
+create(id)
+delete(id)
+connect(edgeID,sourceID,targetID)
+disconnect(edgeID)

Watch

Display
+stringFields: list
+setDisplayField(index,text)

Button
+type: string
+isActive: bool

Indiglo
+isLightOn: boolean
+setLightOn()
+setLightOff()

+addMonth()
+addYear()
-getDaysInMonth()
+setEditBlink(index)
+toggleEditBlink(index)

Button Behaviour

Initial
T: create
A: self = eventhandler.get_event_params()

ButtonOff

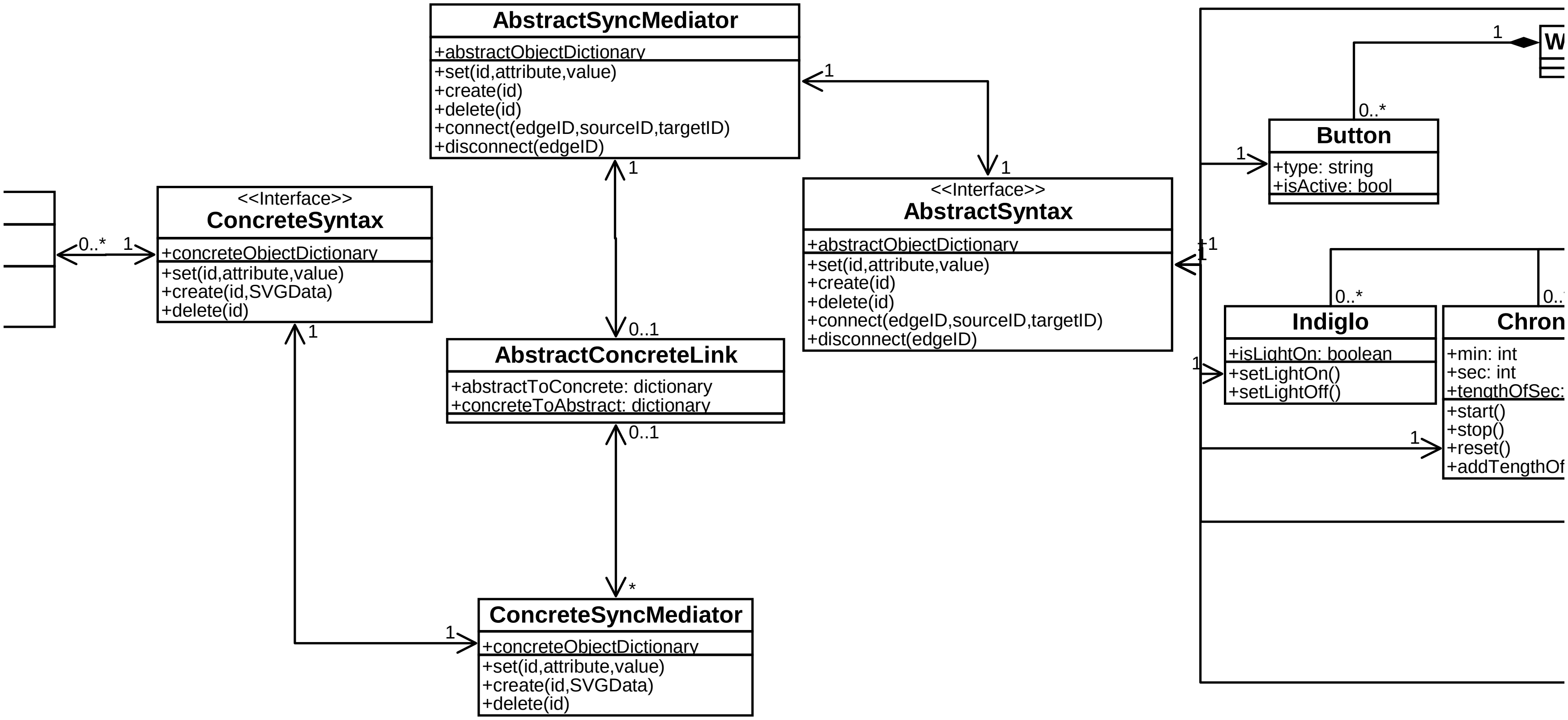
ButtonOn

T: press
G: self.abstract.set(self.type, 'isActive', True)
A: self.watch.event(self.type + 'press')

T: release
G: self.abstract.set(self.type, 'isActive', False)
A: self.watch.event(self.type + 'release')

Abstract To Concrete

- UML Class Diagrams



AbstractSyncMediator

+abstractObjectDictionary

+set(id,attribute,value)
+create(id)
+delete(id)
+connect(edgeID,sourceID,targetID)
+disconnect(edgeID)

<<Interface>>
ConcreteSyntax

+concreteObjectDictionary

+set(id,attribute,value)
+create(id,SVGData)
+delete(id)

<<Interface>>
AbstractSyntax

+abstractObjectDictionary

+set(id,attribute,value)
+create(id)
+delete(id)
+connect(edgeID,sourceID,targetID)
+disconnect(edgeID)

AbstractConcreteLink

+abstractToConcrete: dictionary
+concreteToAbstract: dictionary

ConcreteSyncMediator

+concreteObjectDictionary

+set(id,attribute,value)
+create(id,SVGData)
+delete(id)

W

Button

+type: string
+isActive: bool

Indiglo

+isLightOn: boolean

+setLightOn()
+setLightOff()

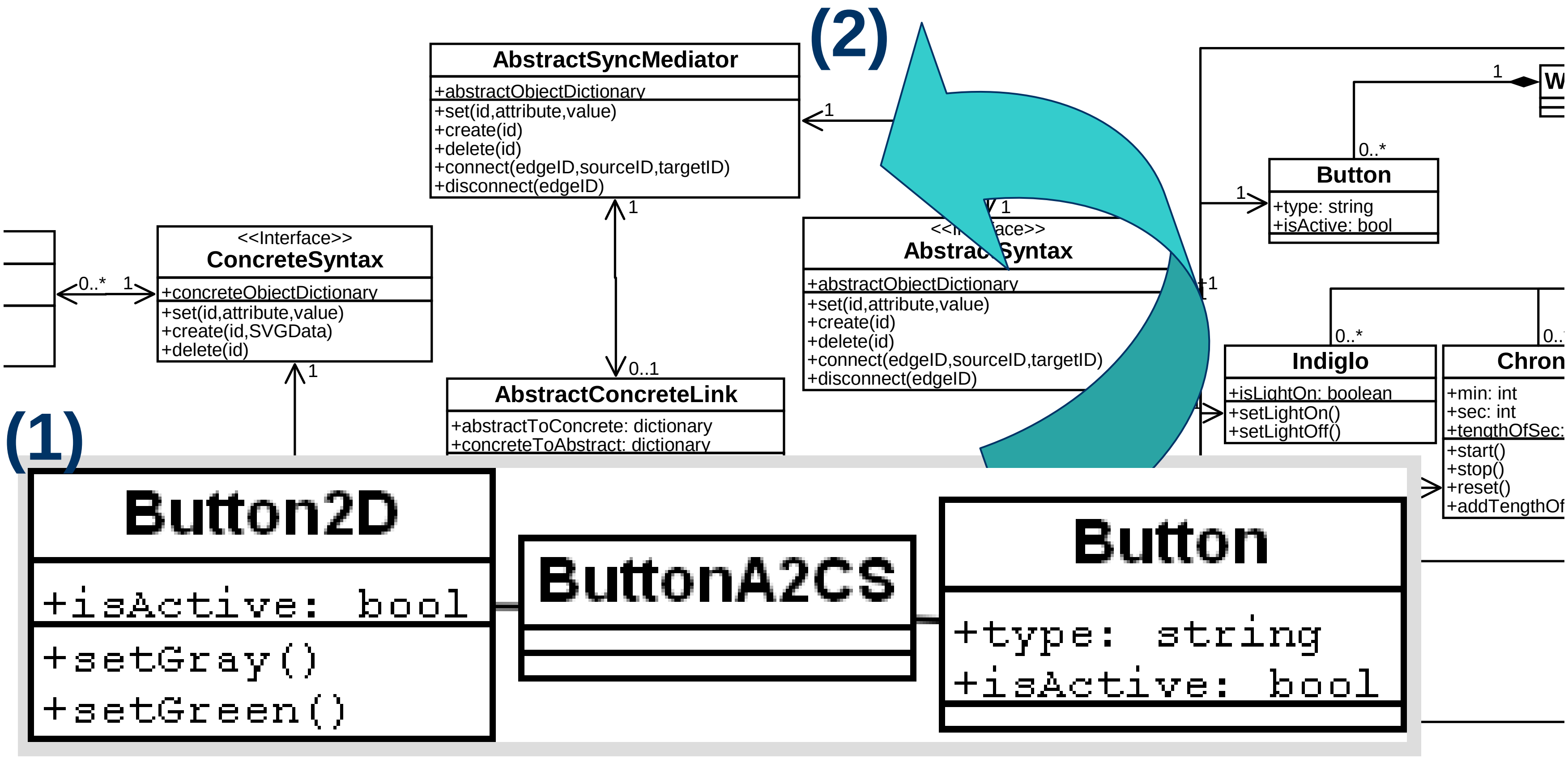
Chron

+min: int
+sec: int
+tengthOfSec:

+start()
+stop()
+reset()
+addTengthOf

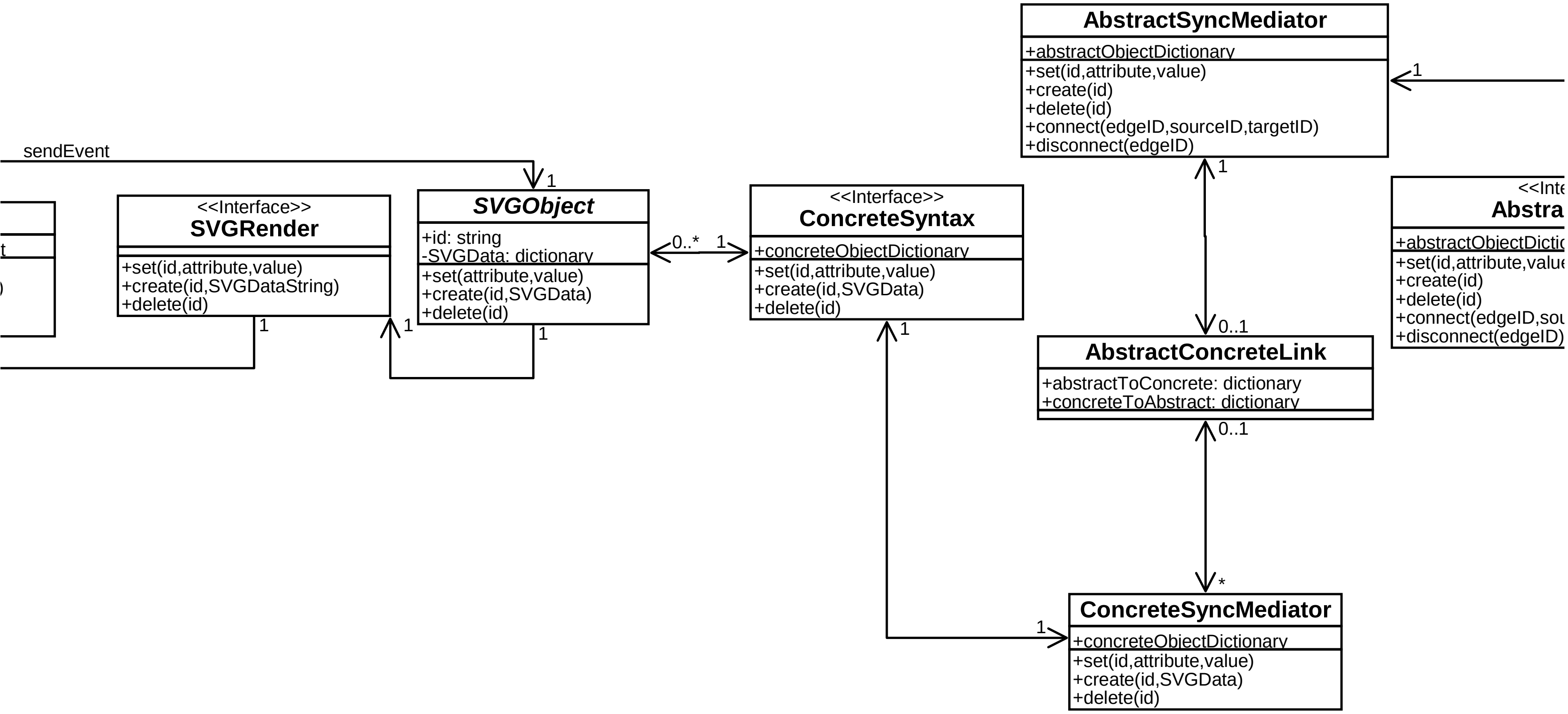
Abstract To Concrete

- UML Class Diagrams



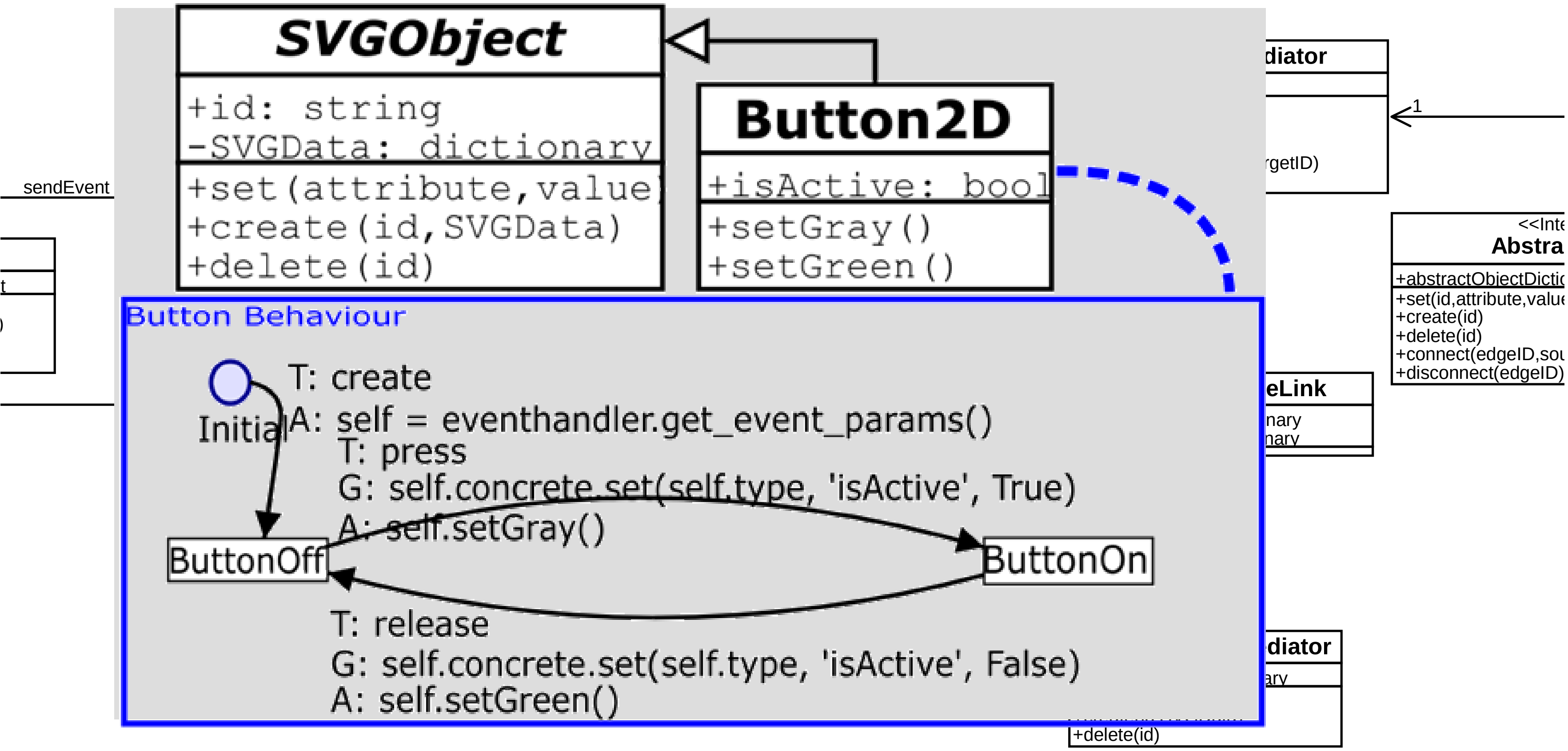
Concrete Syntax 1

- UML Class Diagrams and State Charts



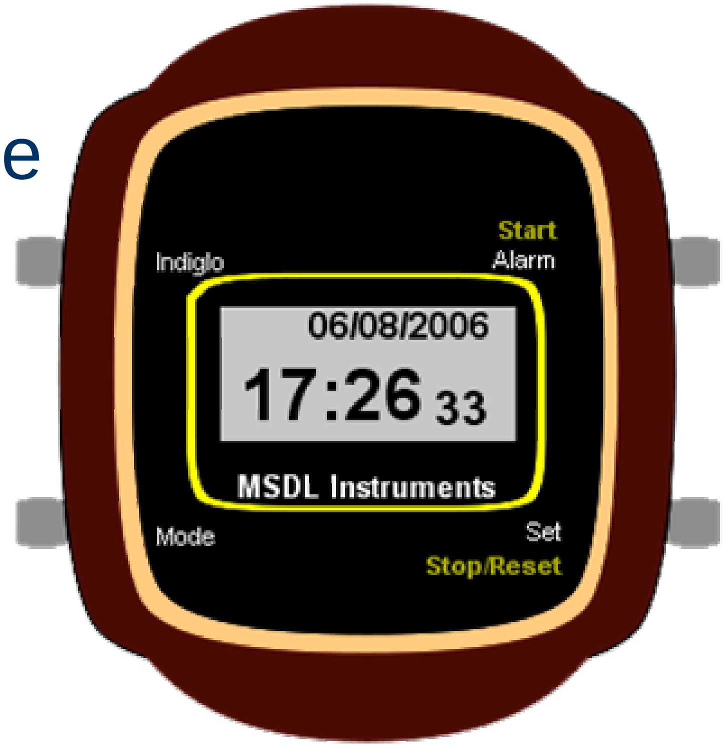
Concrete Syntax 2

- UML Class Diagrams and State Charts



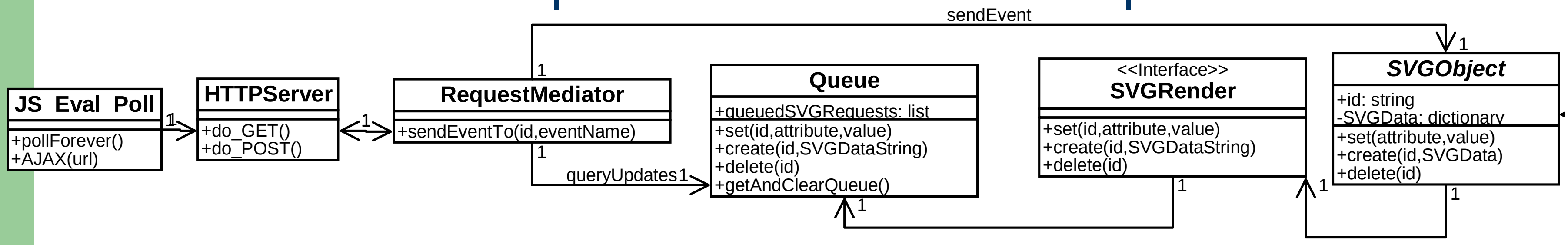
Concrete Syntax 3

- The SVGObject contains the necessary attribute-value pairs for the rendering device to draw the concrete visual representation



AJAX

- Objects updated by abstract changes may queue visual updates
  - The web browser polls constantly with XMLHTTPRequests to obtain these updates



- Events in the web browser are sent by XMLHTTPRequest to the corresponding concrete object