Project presentation #2

Provide behaviour to XML/SVG elements

Fabrice Hong
Informatique semestre 8

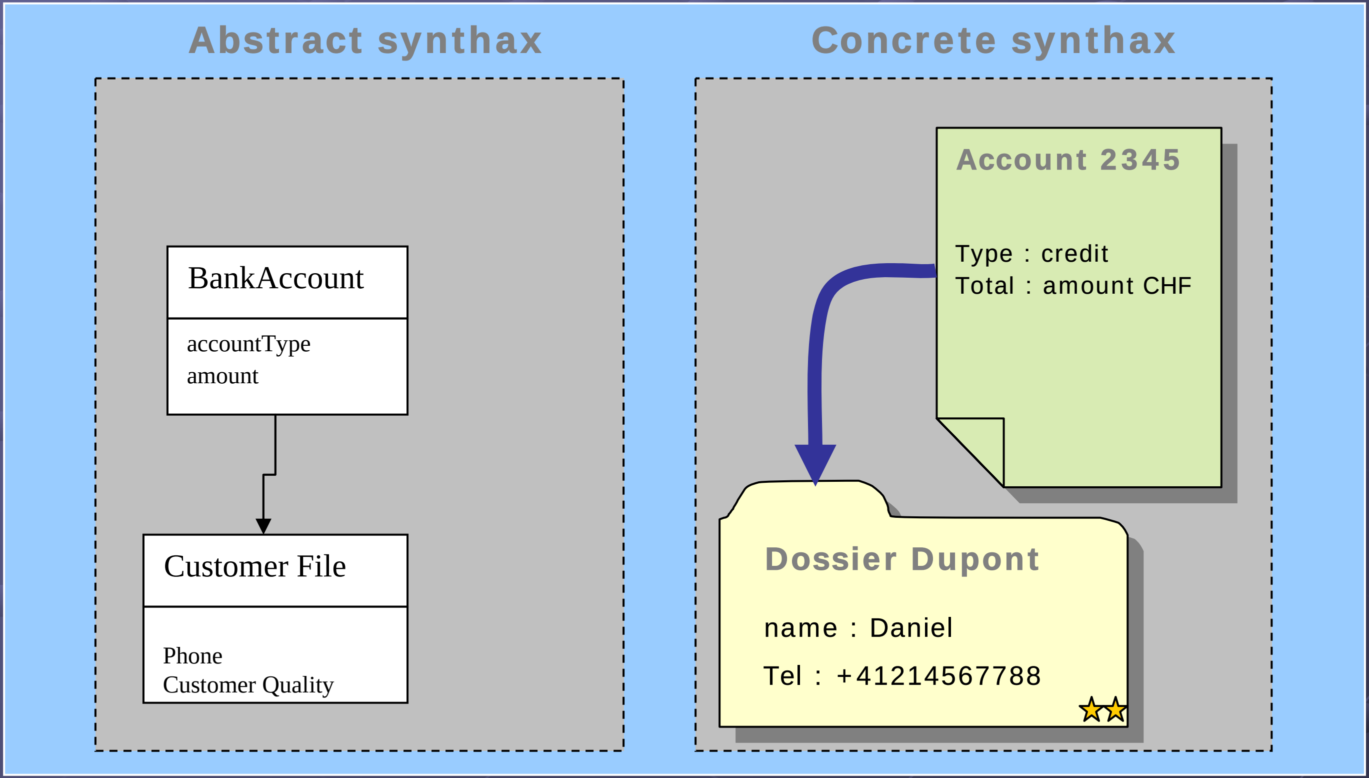
Assistant :
Frédéric Fondement

Provide behaviour to SVG

- 1 - Quick reminder
- 2 - What has been done
- 3 - Next things to do

1 - Quick reminder
Context

- Concrete synthax is the instance of the meta-model

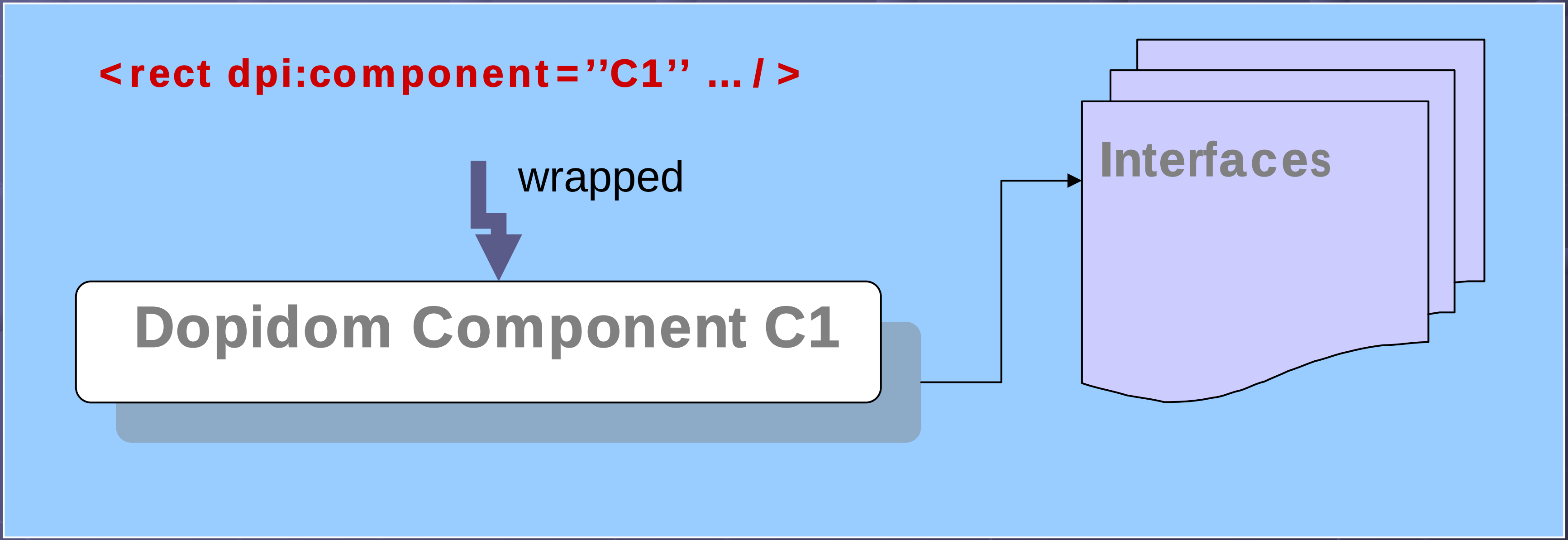


1 - Quick reminder
What we want to do

- Represent a graphical concrete synthax using SVG
- Define models of components
- Give behaviours to components
  - Translation
  - Editiion
  - Ability to link themselves
  - Ability to display informations

1 - Quick reminder
Interfaces

- DOPIDOM propose an architecture to implement SVG component behaviours by mean of interfaces
- Interface can consume actions or queries
- A dopidom component can have several interfaces
- We assign dopidom component to SVG graphics



Provide behaviour to SVG

- 1 - Quick reminder
- 2 - What has been done
- 3 - Next things to do

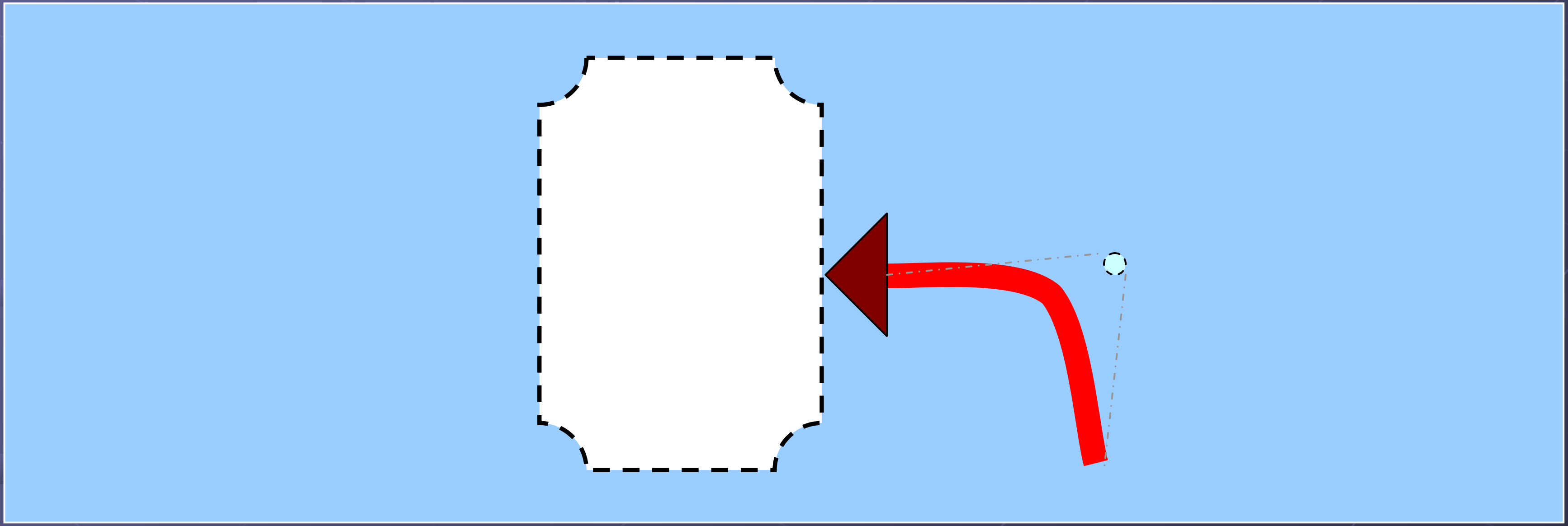
2 - What has been done

- 2.1 - Component : Links
- 2.2 - Component : Graphic Container
- 2.3 - Constraints
- 2.4 - Grouping Elements

2 - What has been done

Component : Links (1)

- Involved components
  - Anchor Points
  - ArrowStart / ArrowEnd
  - Curved Line
  - Line handle
  - Link



2 - What has been done
Component : Links (2)

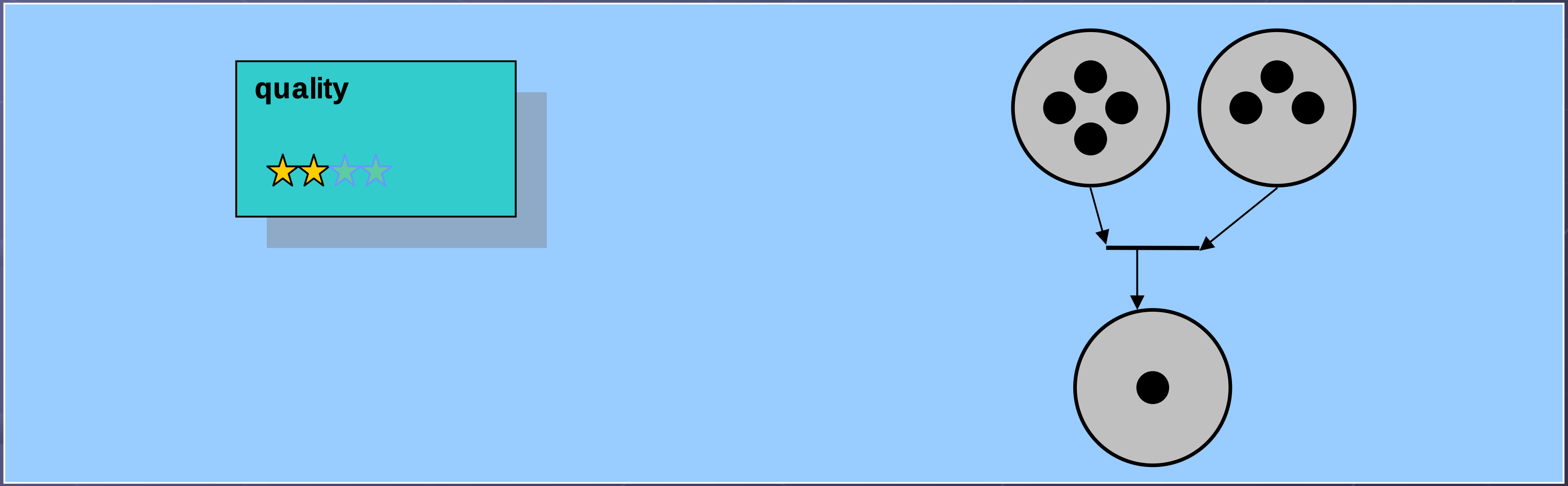
- How to use links
  - Define AnchorPoints
  - Define link properties
    - Color, stroke width, arrow types
  - Define Arrows
- SVG definition
  - <circle dpi:component="AnchorPoint" linkType="#link1" r="30" .../>
  - <g id="link1" dpi:component="Link" arrowStart="#arrow" arrowEnd="#arrow" stroke="blue" stroke-width="2"/>
  - <polygon id ="arrow" dpi:component="Arrow" ax="15" ay="0" points="..." width="30" .../>

🟥 Standard SVG
🟧 extensions
🟨 important

2 - What has been done

Component : Graphic Containers (1)

- Used to translate numeric information to graphical ones
- Involved Components
  - GraphicContainer
  - ContainedGraphic
- Different positionement methods



Provide behaviour to SVG elements

2 - What has been done

Component : Graphic Containers (2)

- How to use it
  - Chose the type of element disposition you want
  - Define the GraphicContainer
  - Define the Contained element
- SVG definition

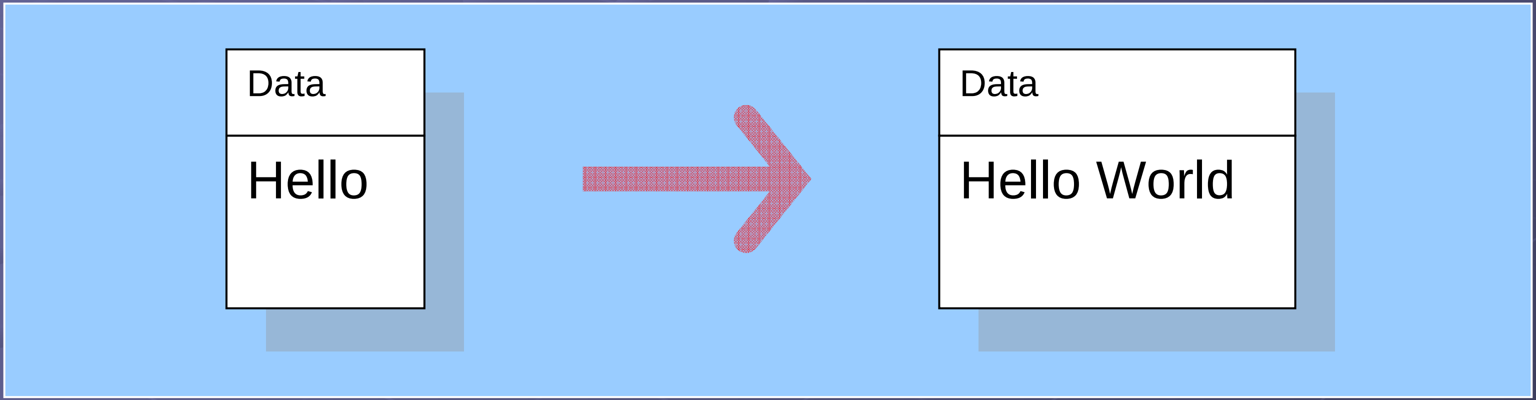
```
<circle dpi:component="GraphicContainer" cx="0" cy="0" r="30" stroke-
   width="2" stroke="black" fill="white"  graphic="#cg1"/>

<circle dpi:component="ContainedGraphic" id="cg1" cx="0" cy="0"
   r="4" fill="black"/>
```

🟥	Standard SVG
🟧	extensions
🟨	important

2 - What has been done

Constraints



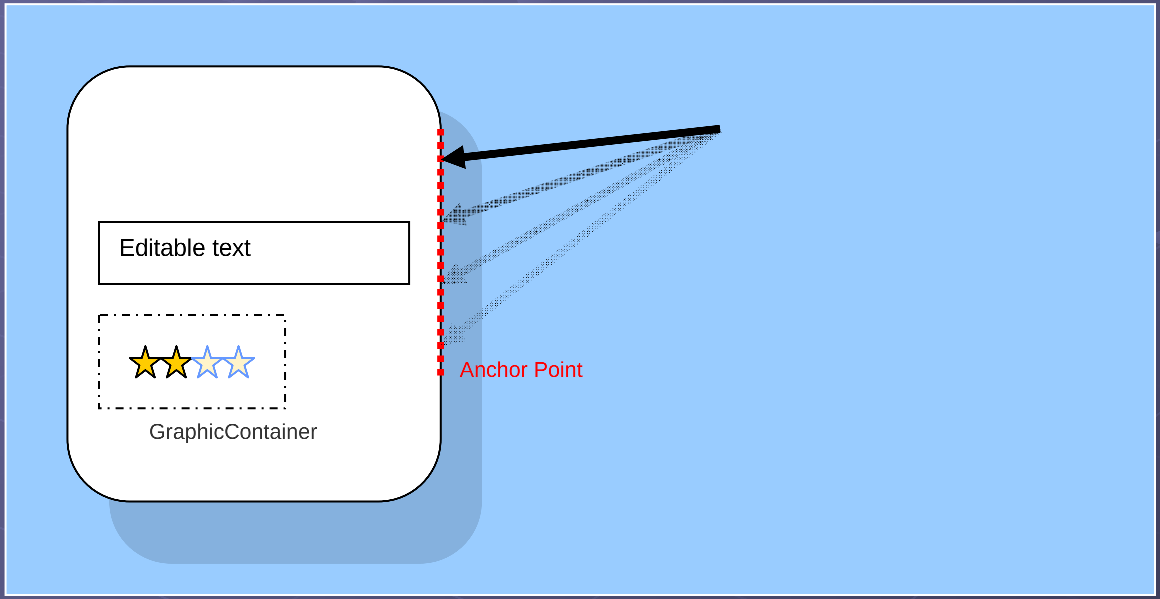
- We want certains SVG attributes to be restricted by calculated values
- We use a toolbox named CSVG
  - `<c:variable name="w" value="c:width(c:bbox(id('text1'))) + 10"/>`

  -
    ```
    <rect width="15" ...>
      <c:constraint attributeName="width" value="$w"/>
    </rect>
    ```

🟥 Standard SVG
🟧 extensions
🟨 important

2 - What has been done

Grouping Elements (1)

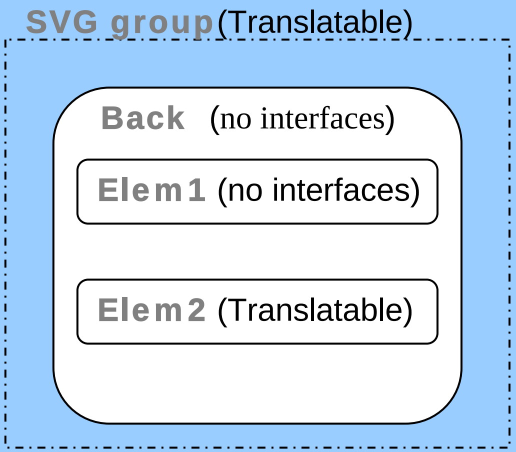


- Sometime we want to design multiples component objects
- Two ways to make the component stay togheter
  - Use interface « Stickable » (dopidom level)
    - Usable « online »
  - Use SVG groups (SVG level)
    - Inheritance of properties

2 - What has been done

Grouping Elements (2)

- Inheritance of property with SVG groups



```
<g>
    <Back …/>
    <Elem1 …/>
    <Elem2 …/>
</g>
```

Incoming Translate Action on…

- **Elem1** or **Back**
  - All elements in the group move
- **Elem2**
  - Only Elem2 moves

Provide behaviour to SVG

- 1 - Quick reminder
- 2 - What has been done
- 3 - Next things to do

Next things to do

- Enhance links
  - Possibility to add points
  - Possibility to set other line types
- State saving
  - All data in SVG components must be saved in to SVG
- Enhance behaviour assignement
  - <u>Now :</u> assign dopidom component to SVG elements
  - <u>Future :</u> possibility to assign singles interfaces
- Refactoring