UI Development using Statecharts

Detlev Van Looy

Universiteit Antwerpen

About me

- Detlev.VanLooy@student.ua.ac.be
- 1st Ma Informatica at University of Antwerp

Overview

- Introduction
- Summary “Rapid development of scoped user interfaces” [1]
- My work
- Result
- Conclusions
- Questions

[1] Denis Dubé, Jacob Beard, H. Vangheluwe, 2009. Rapid development of scoped user interfaces

Introduction(1)

- Development of complex User Interfaces
- Many components with different behaviour and relations
- UIs should be easy to maintain
- Code-centric implementations are not adequate

Introduction(2)

- Try to minimize “accidental complexity” [2]
- Model every aspect of the system at the most appropriate level of abstraction
- Use Hierarchically-linked Statecharts to model a Scoped User Interface
- In this case a UI for statecharts

[2] Brooks, F., 1987. No silver bullet: Essence and accidents of software engineering.

Summary(1): intro

- As said in the introduction of this presentation
- Need to facilitate rapid, domain-specific modelling of a UI
- Achieve this by modelling the behaviour of each individual UI component

Summary(2): Scoped UIs

- UI where visual components (buttons/windows/entities) are hierarchically nested
- Top level is more general behaviour
- Deeper levels are more specific behaviour
- A scoped UI can bind an event to the most tightly-binding component in a hierarchy

Summary(3): Scoped UIs

- Focus on domain/formalism-specific modelling environments, these can improve productivity as they:
  - Match the user’s mental model
  - Constrain the user to the problem at hand
  - Separate domain-expert’s work from that of others
  - Can exploit features inherent to a specific domain/formalism

Summary(4): Scoped UIs

- Two challenges when developing Scoped UIs:
  - How to describe interaction between user and entities of the UI

    => consider the entities as actors

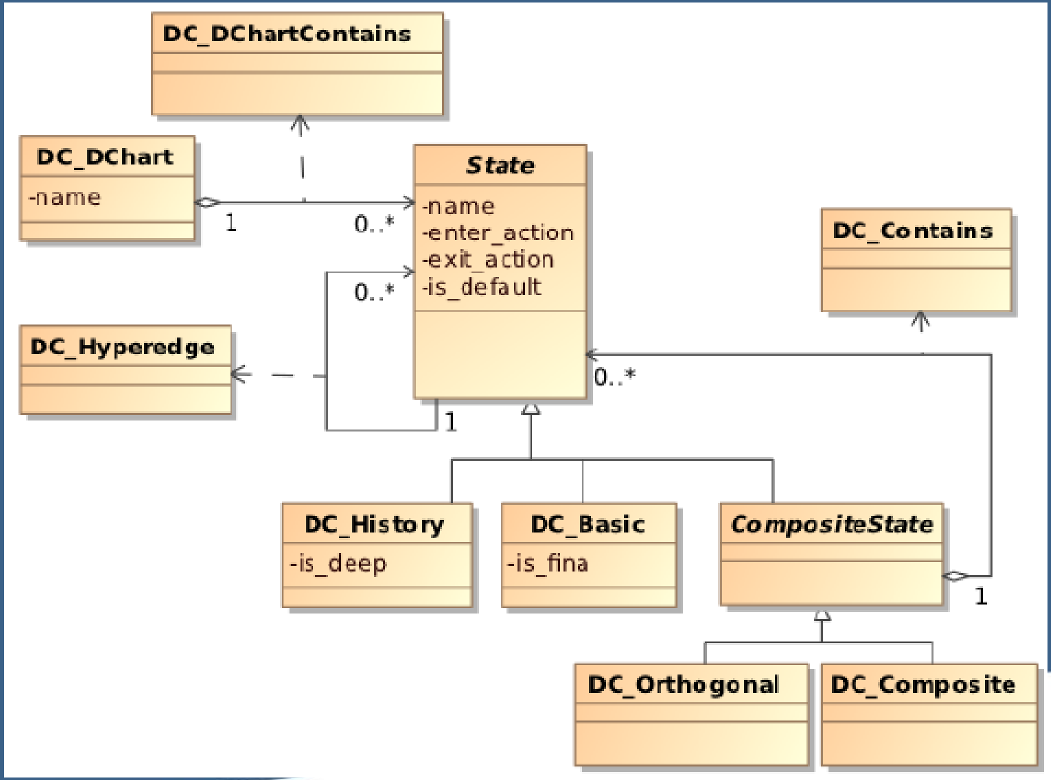
  - Dont create new specification of UI behaviour for each formalism
    => have a generic specification at the root level

Summary(5): HIS

- Hierarchically-linked Statecharts is a formalism for visually describing Scoped UIs
- Workflow:
  - Specify abstract syntax (for example using UML Class Diagrams)
  - Model concrete visual syntax (associate a visual entity)
  - Specify UI behaviour using Statecharts, each Statechart associated with a class, specifying the reactive behaviour of that class

Summary(6): Example

- Specify abstract and concrete syntax



Summary(7): Example

- DC_Dchart represents the entire model. All other entities are contained by this
- DC_Basic, DC_Composite, DC_History, DC_Orthogonal, …
- Should all be familiar from using statecharts for the Digital Watch assignment in MoSIS

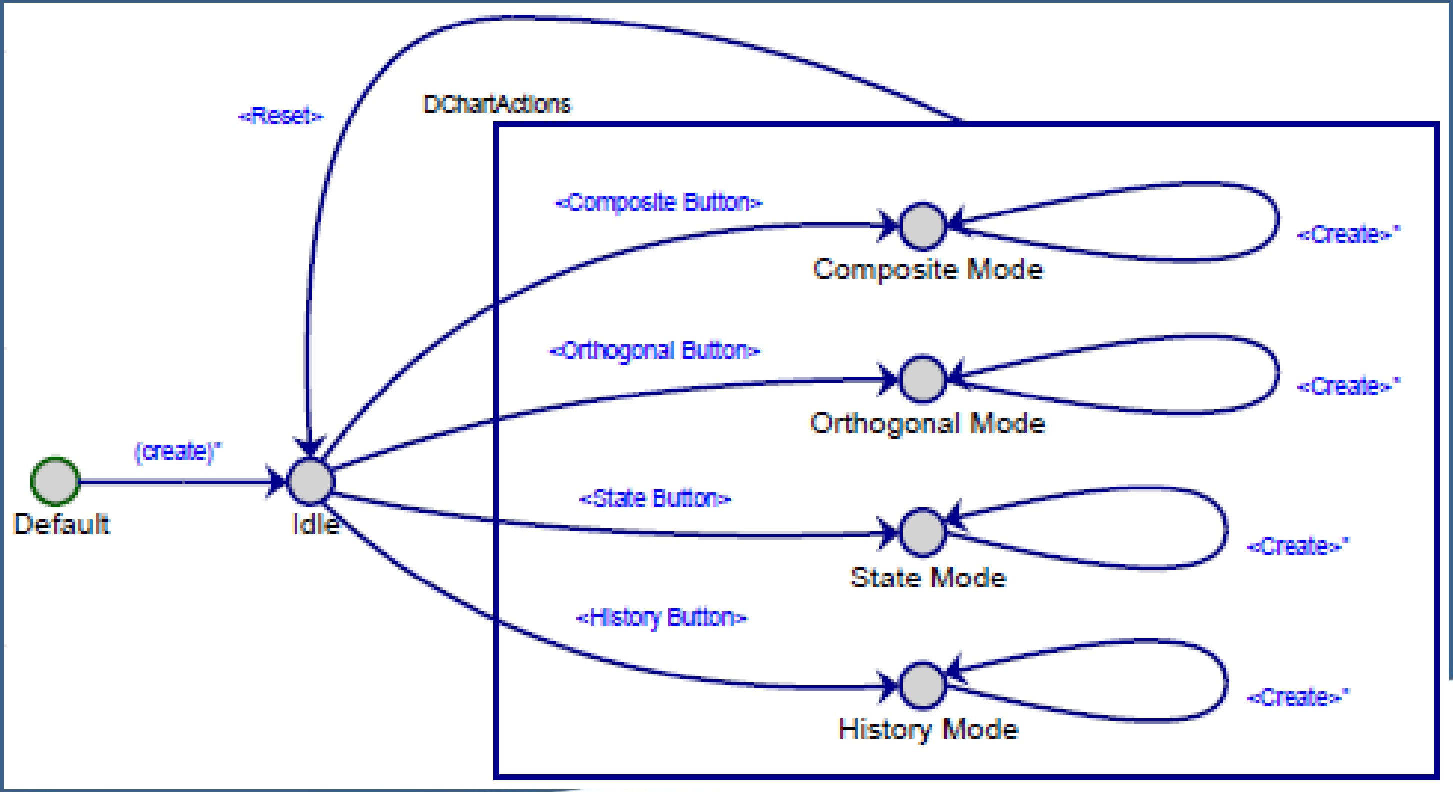
Summary(8): Example

- Specify formalism-specific behaviour
- Some notes on event labels:

x*	action code is present
x+	a different statechart handles the action
<x>	event generated by another statechart
(x)	initialization routine
[x]	event generated by the statechart itself

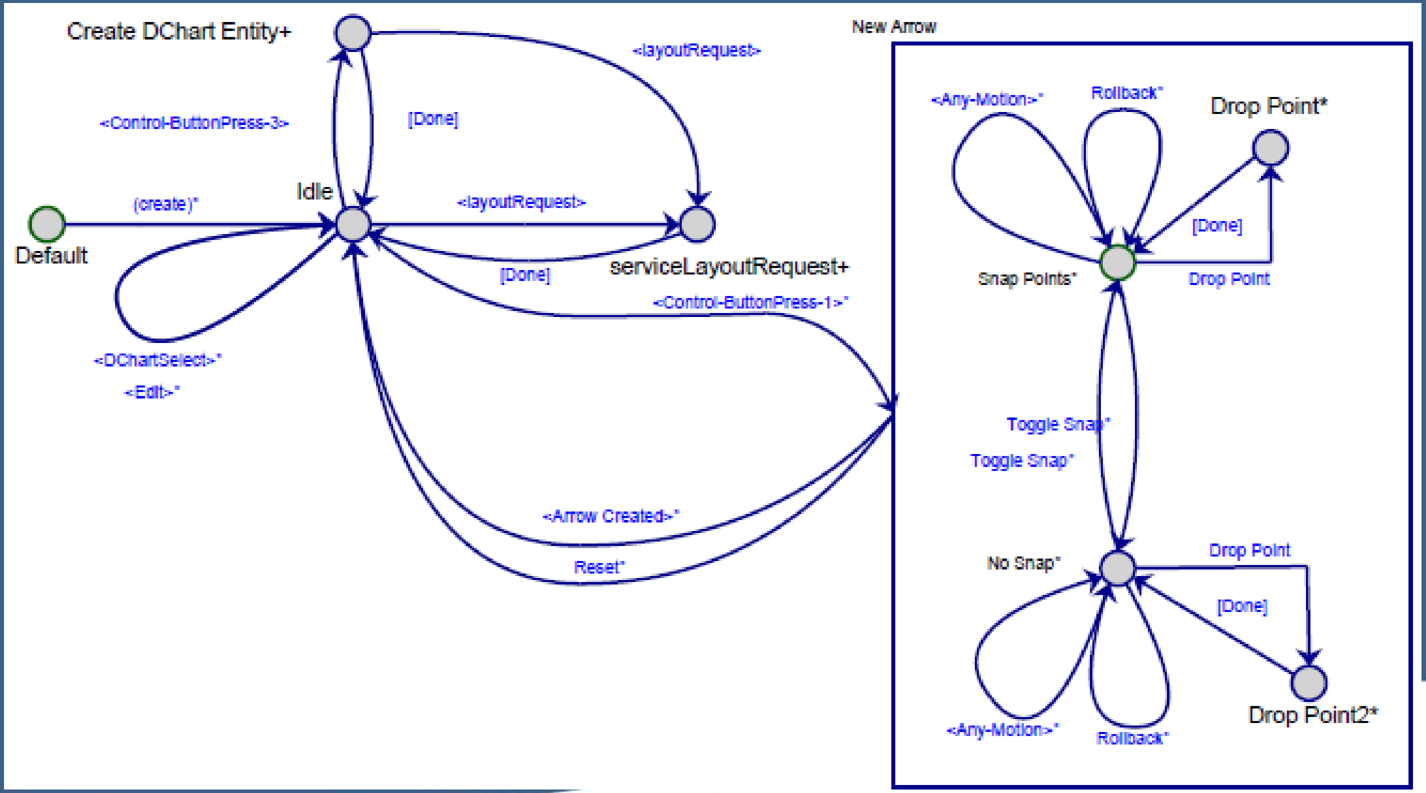
Summary(9): Example

- Button Behaviour



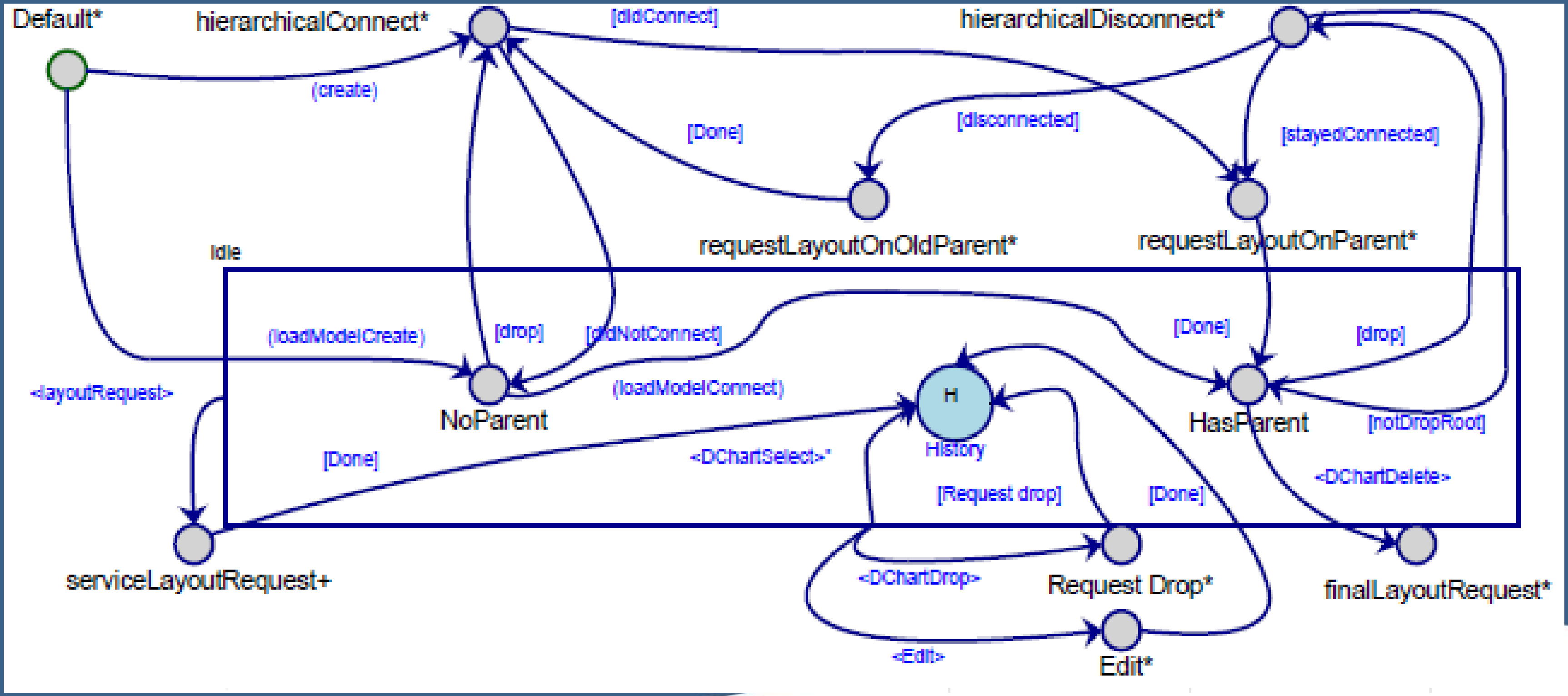
Summary(10): Example

- DC_Dchart behaviour



Summary(10): Example

- DC_Composite behaviour

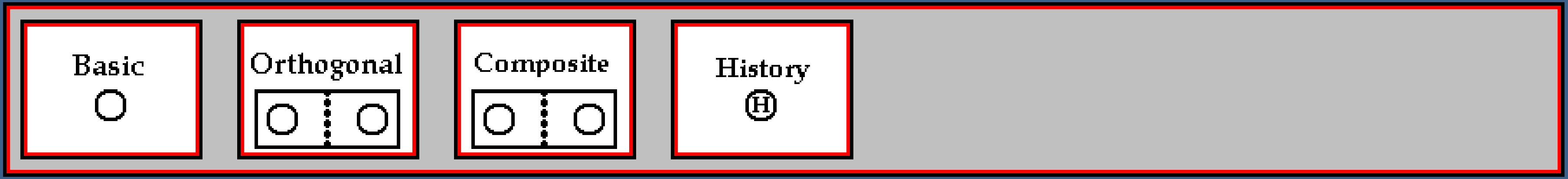


Summary(11): Conlusion

- Given later together with my conclusions

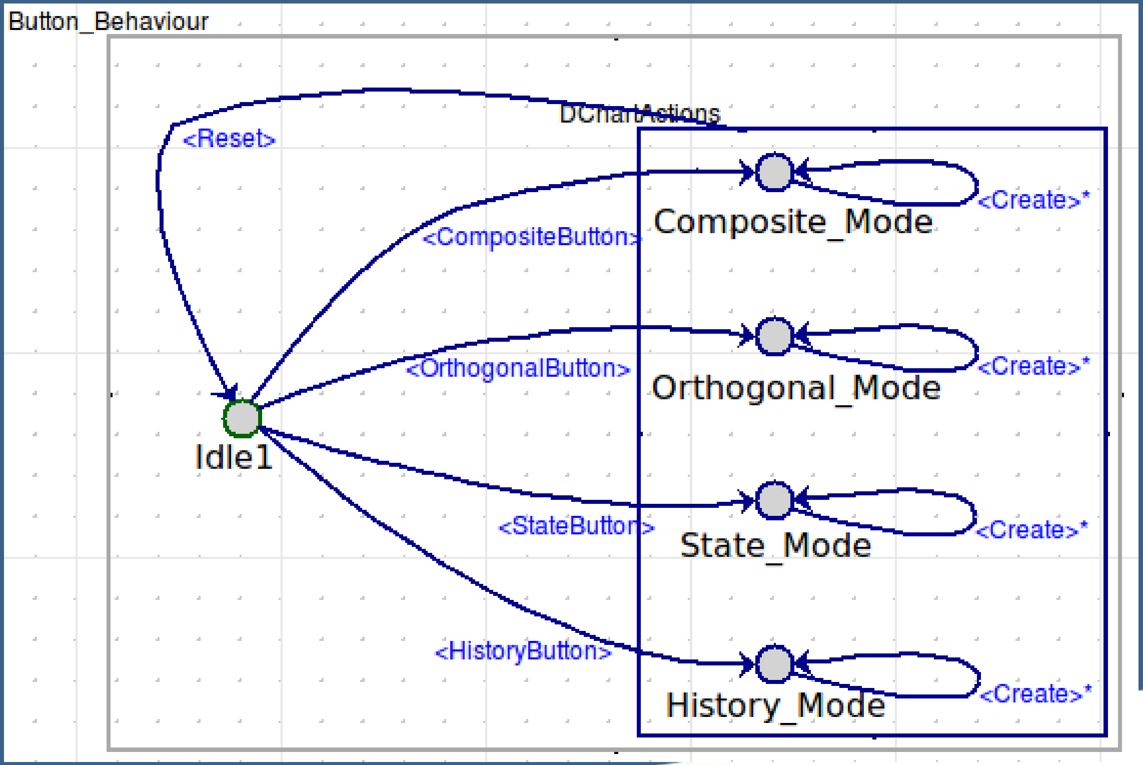
My work(1)

- Explained in detail, one behaviour at a time
- But first some introduction
- My buttons menu:



My work(2)

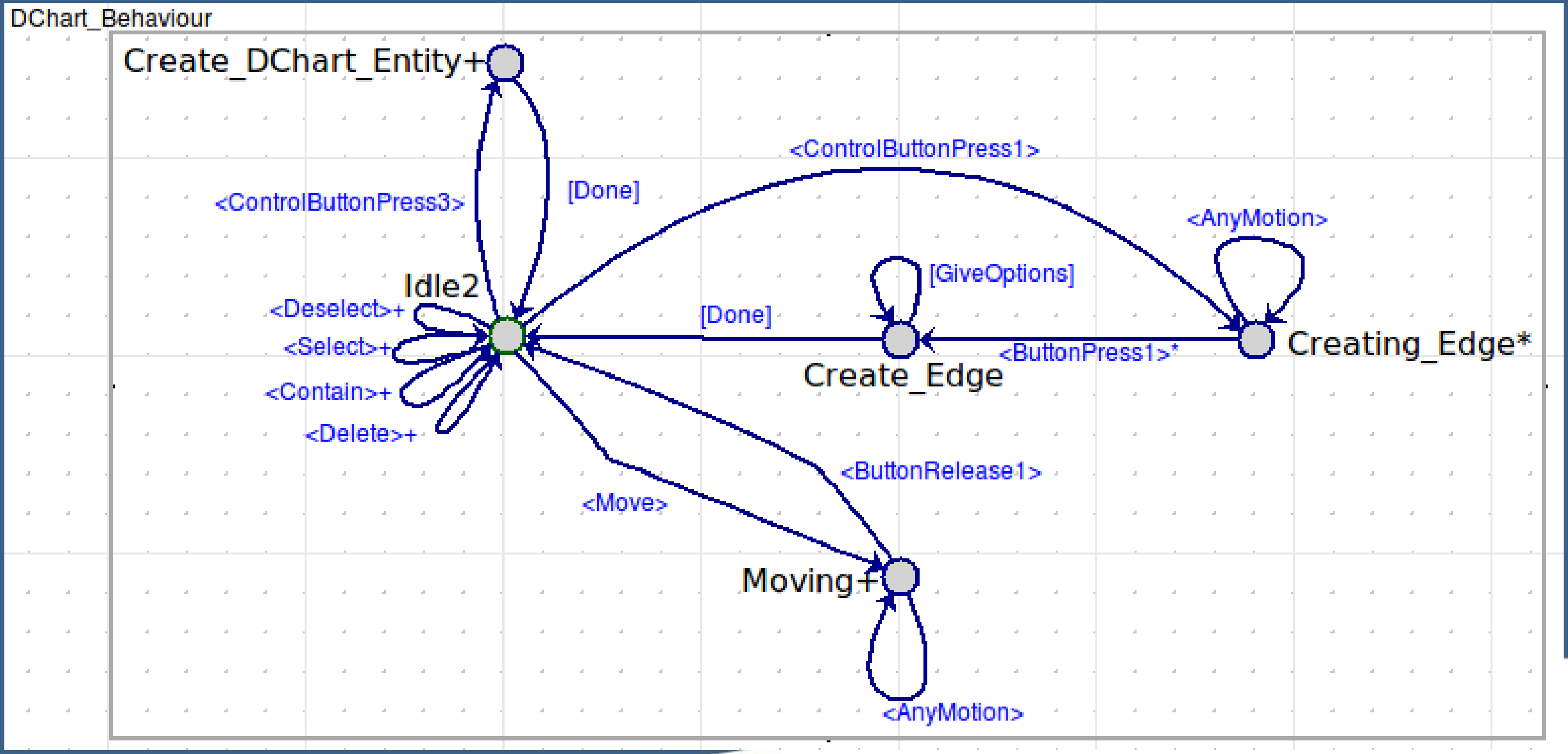
- Button_Behaviour (same as original)



Universiteit Antwerpen

My work(3)

- DChart_Behaviour



My work(4)

- Creation of states
  - Ctrl+right clicking on the canvas
    => ControlButtonPress3 event to DC_Dchart
    => Create event to Button behaviour
    => call the corresponding drawXstate() method

My work(5)

- Creation of edges
  - Ctrl+left clicking on a state

    => ControlButtonPress1 event to DC_Dchart

    => go to “Creating_Edge*” state and lock input
    => mouse motion generates AnyMotion event
    => left clicking somewere generates ButtonPress1

    (cont.)

My work(6)

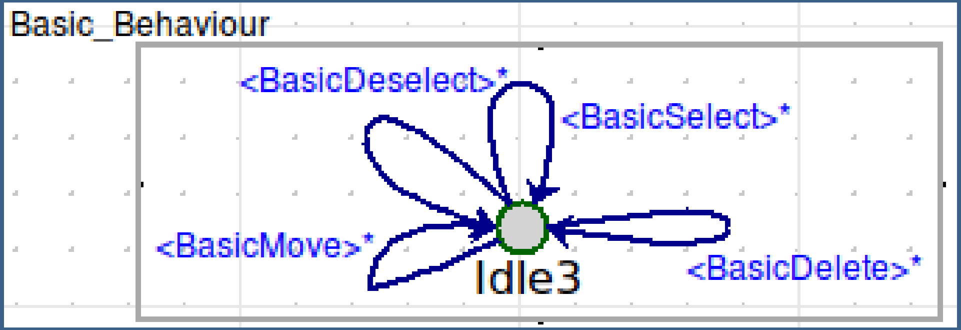
- Creation of edges (cont.)
  - => results in 4 options: “failed”, “transition”, “containment” and “giveoptions”
  - => afterwards a Done event is sent, the input is unlocked and the edge creation is finished

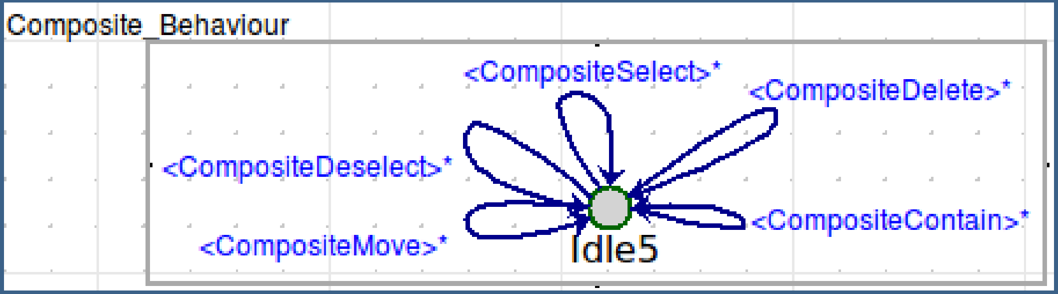
My work(7)

- The following behaviours all follow the same pattern:
  - An action by the user triggers an event in DC_DChart
  - A method is called to check which entity has to do the behaviour
  - An event is sent to the corresponding statechart which executes the behaviour

My work(8)

- Behaviour statecharts





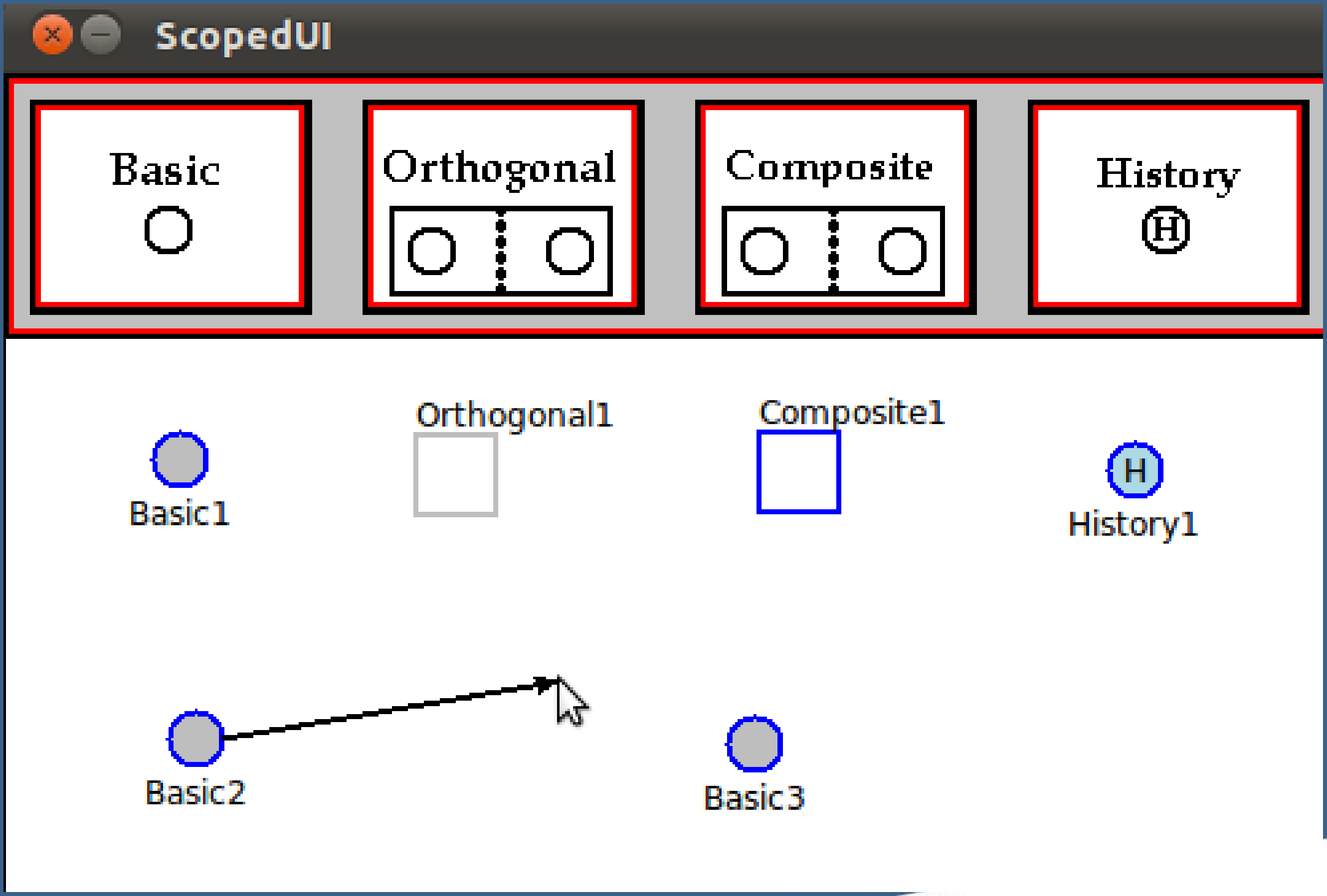
My work(9)

- Possible behaviour: selection, deselection, deletion, containment, movement
- Containment only for composite/orthogonal states
- Movement also takes DC_DChart to state “moving+”, which is a bit like the creation of an edge

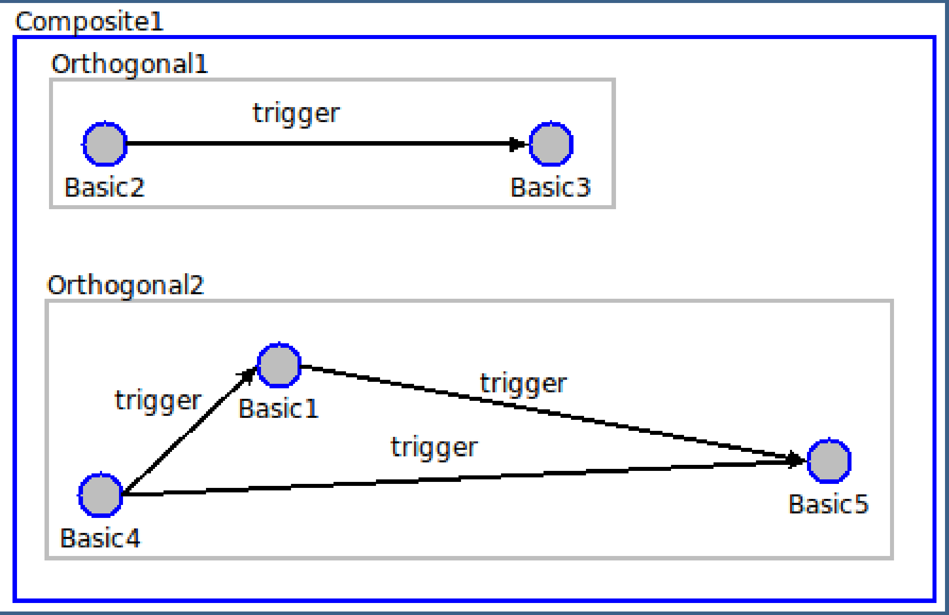
My work(10)

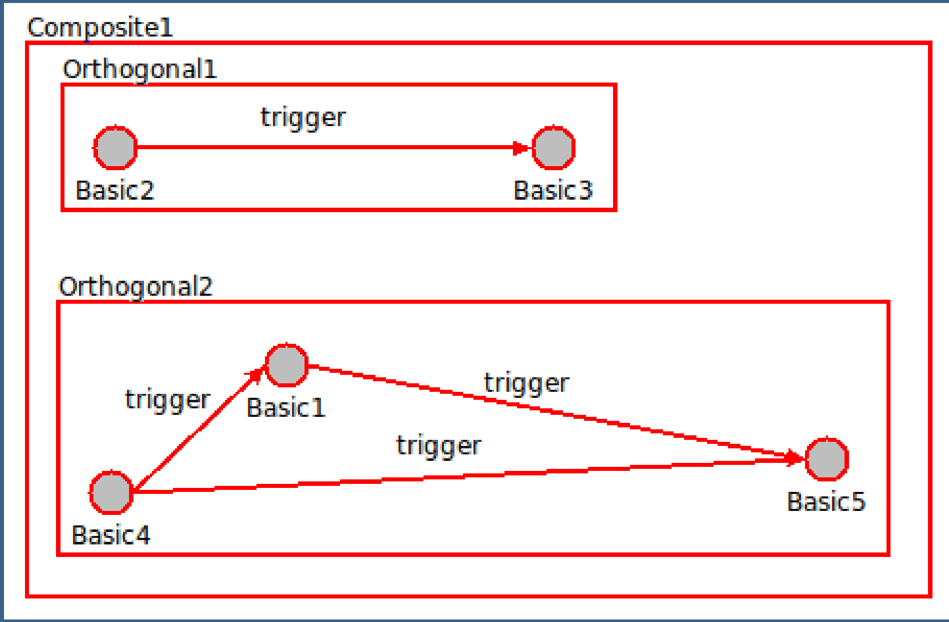
- Example: selection
  - Left click on an entity
    => Select event is sent to DC_Dchart
    => calls method to see which kind of entity was selected
    => sends Select event to corresponding statechart which calls its drawSelect method
  - Orthogonal/composite drawSelect will recursively select every contained entity

Result(1)

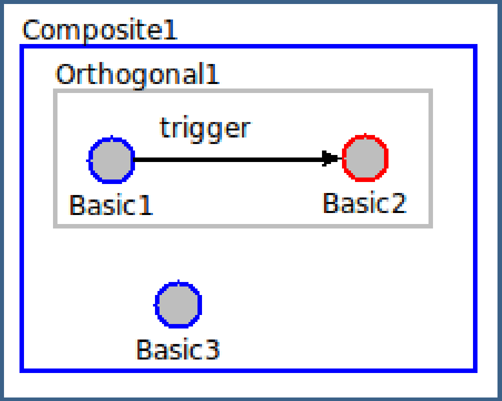

ScopedUI
Basic
Orthogonal
Composite
History
H
Orthogonal1
Composite1
Basic1
H
History1
Basic2
Basic3

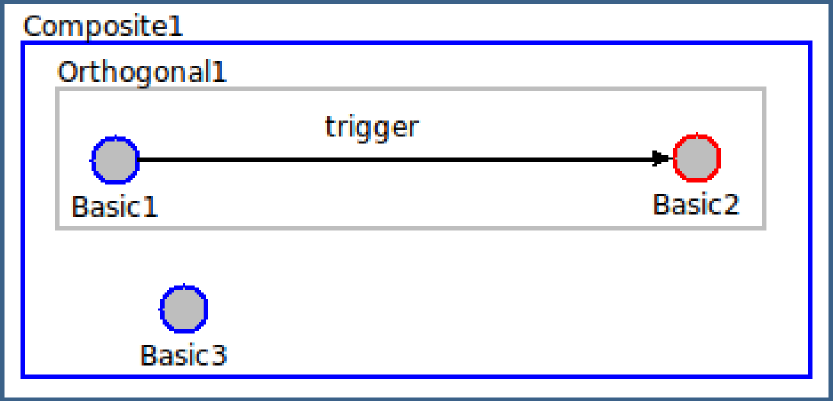
Result(2)


Composite1
Orthogonal1
trigger
Basic2
Basic3
Orthogonal2
trigger
Basic1
trigger
trigger
Basic4
Basic5


Composite1
Orthogonal1
trigger
Basic2
Basic3
Orthogonal2
trigger
Basic1
trigger
trigger
Basic4
Basic5

Result(3)


Composite1
Orthogonal1
trigger
Basic1
Basic2
Basic3


Composite1
Orthogonal1
trigger
Basic1
Basic2
Basic3

Conclusions

- It is possible to model complex, scoped, formalism-specific behaviour using HIS.
- It is possible to develop it quickly
- The implementation is robust and easy to maintain

Questions

- Thank you for your attention!
- Questions?

Universiteit Antwerpen

References

- [1] Denis Dubé, Jacob Beard, H. Vangheluwe, 2009. Rapid development of scoped user interfaces
- [2] Brooks, F., 1987. No silver bullet: Essence and accidents of software engineering.