Building Completely Custom Flex Components

obecto
passionate about software & people

There are many ways...

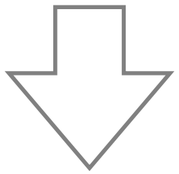
- creating a composite MXML component
- extending an existing component
- extending UIComponent

To extend the UIComponent we need to know the...

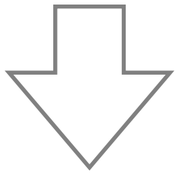
UIComponent Life-cycle

obecto
passionate about software & people

Initialization Phase

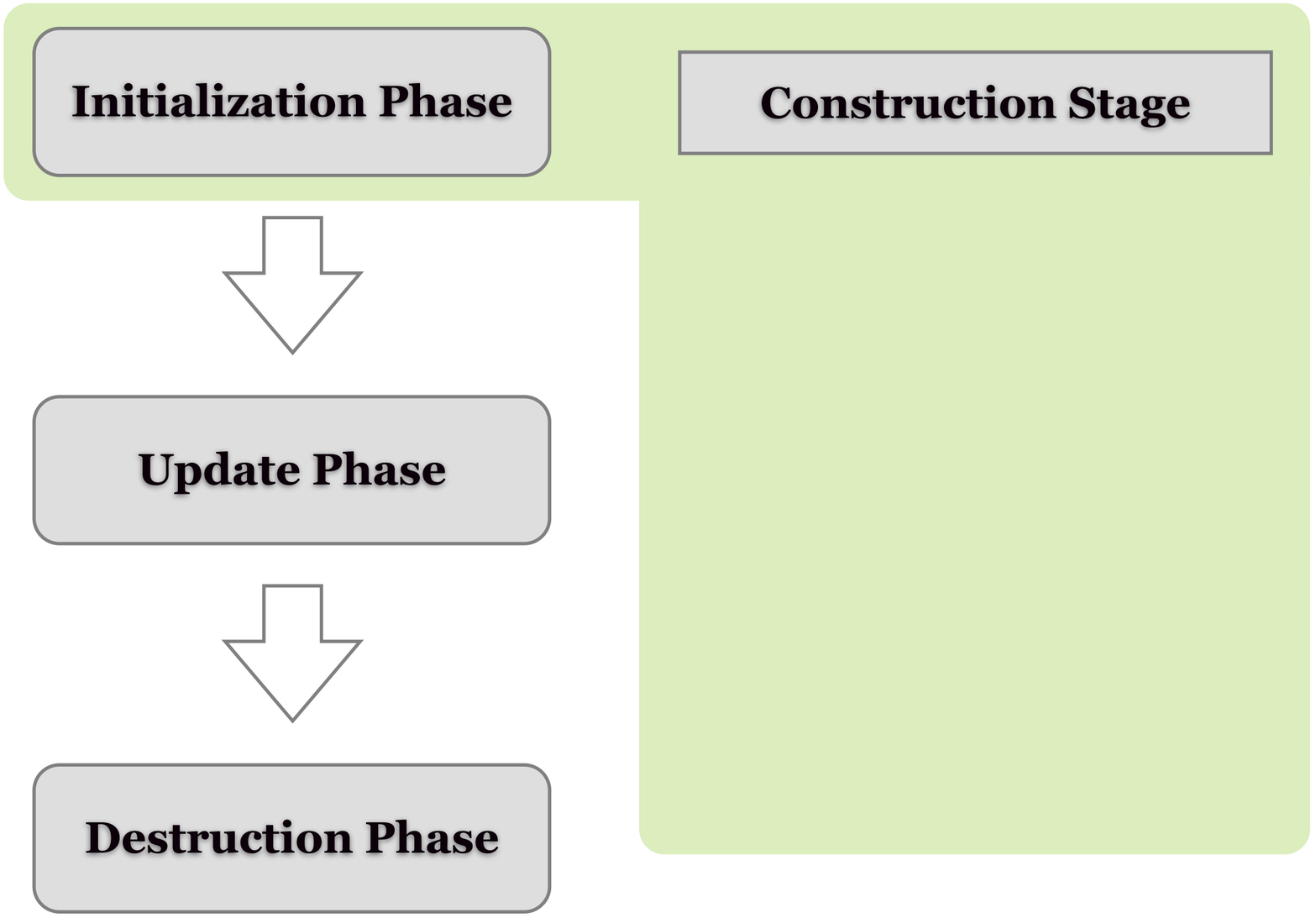


Update Phase

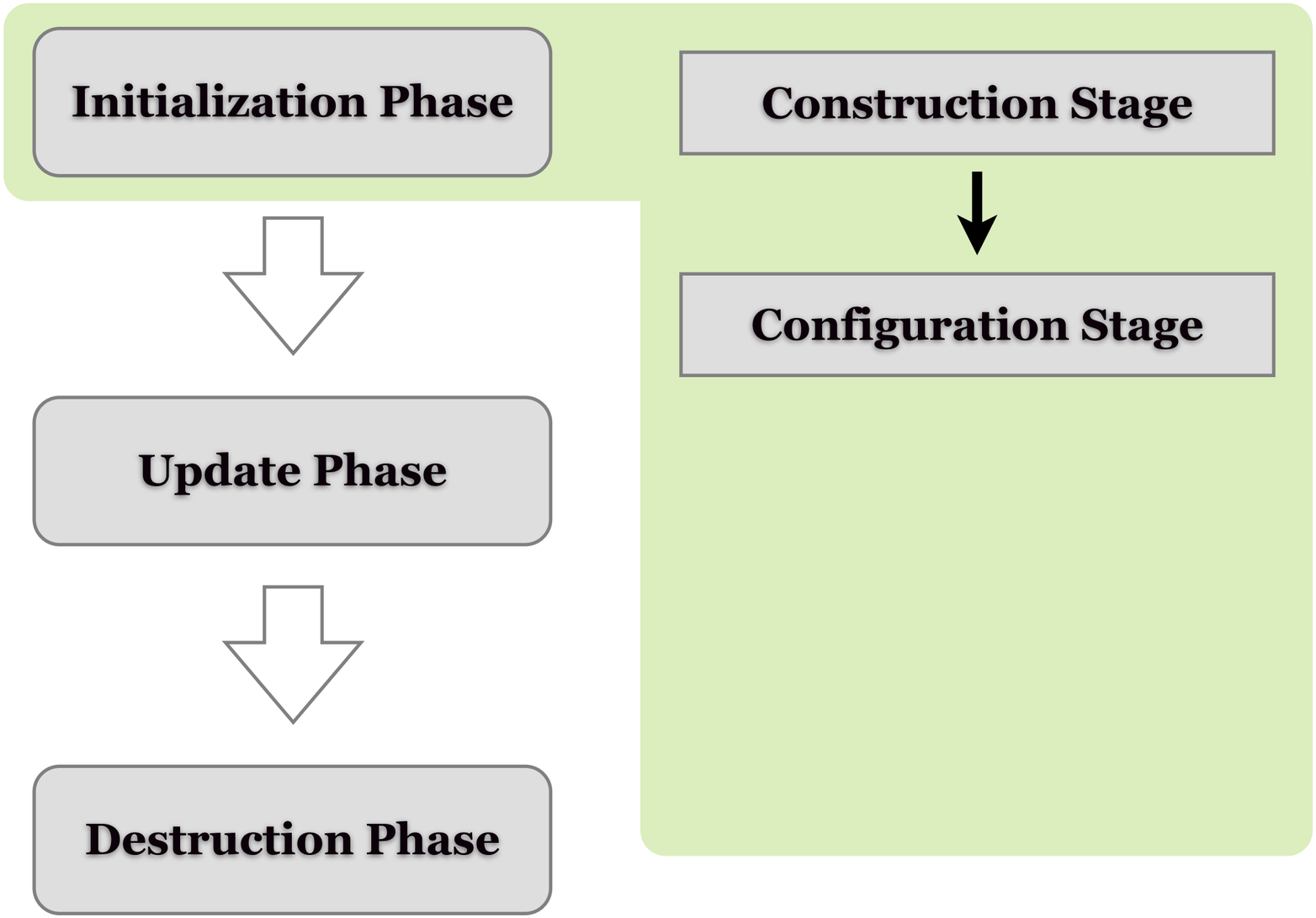


Destruction Phase

obecto
passionate about software & people

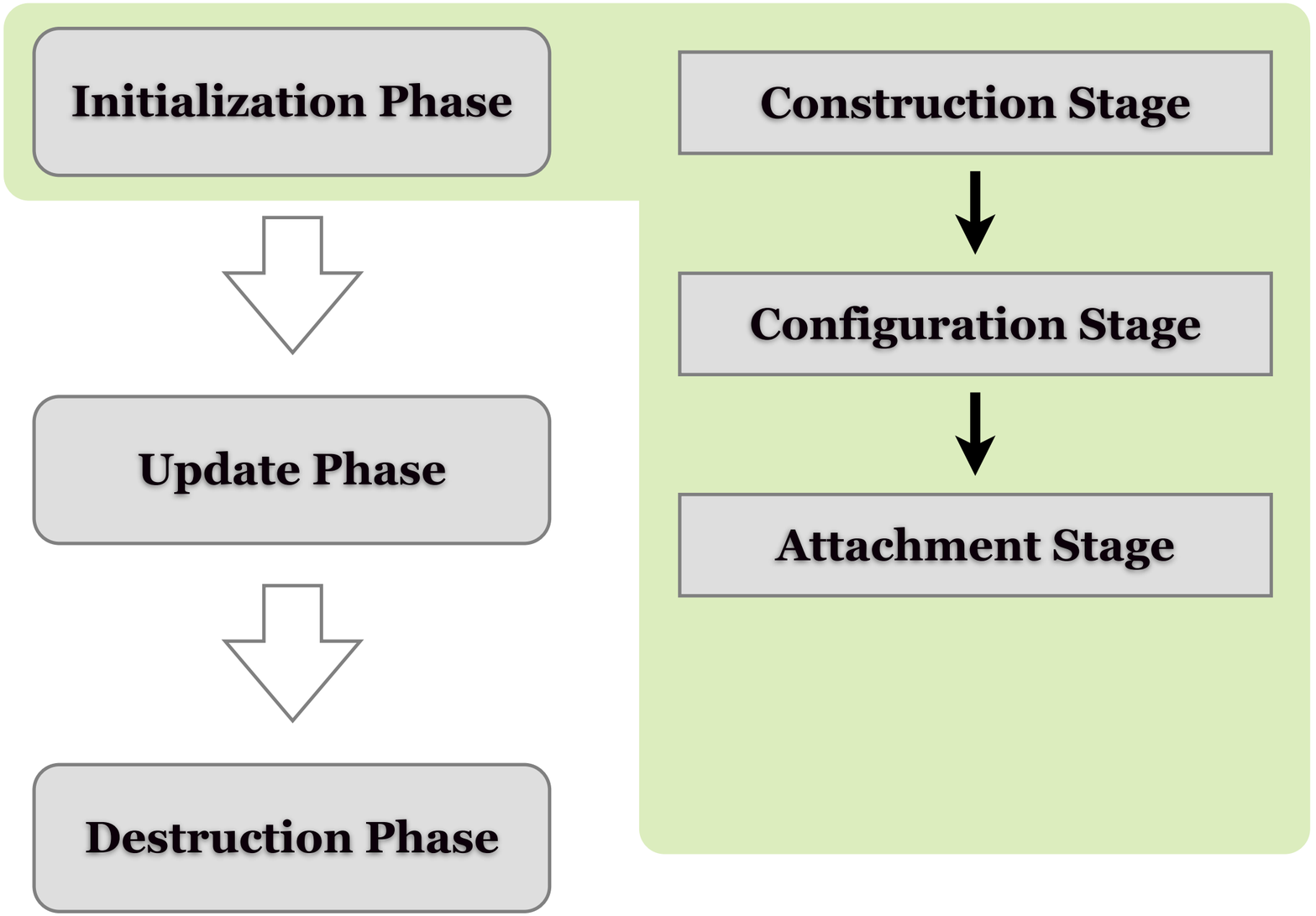


- the constructor is called
- don’t create display objects in the constructor
- set initial values of component properties
- add event listeners
- initialize other objects

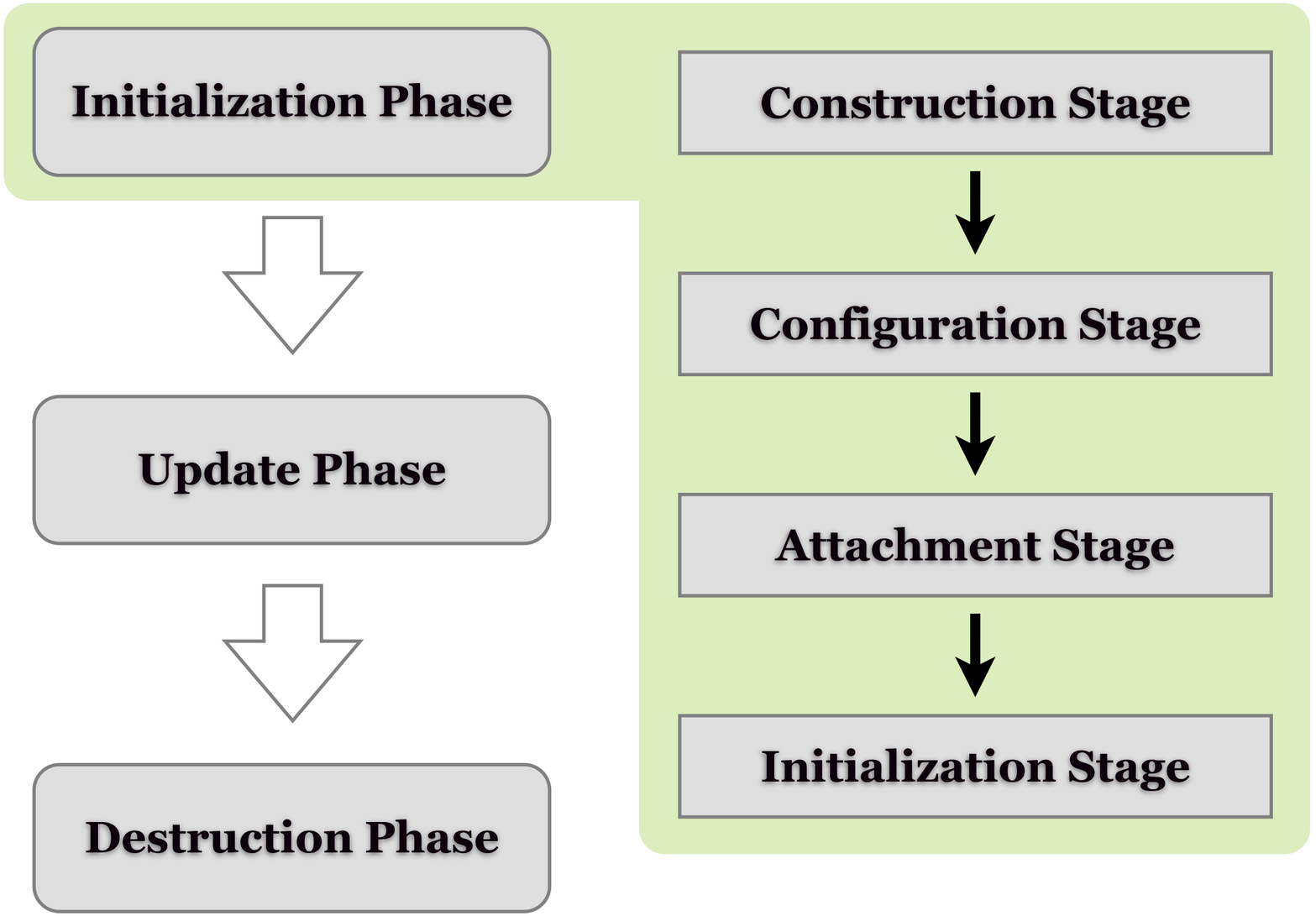


- configuring the newly-constructed instance
- properties
- styles
- event handlers

obecto
passionate about software & people

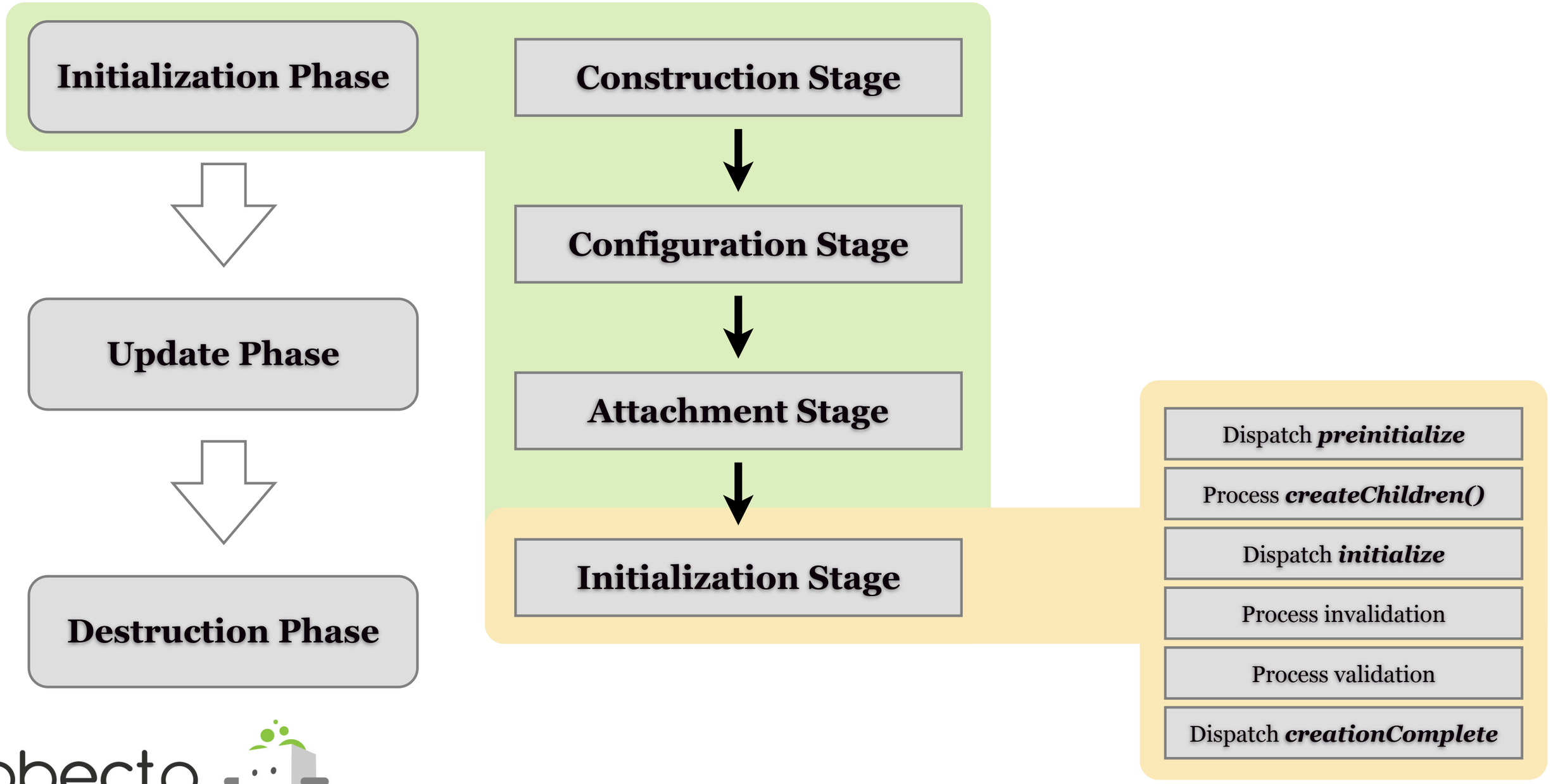


- the component is added to the display list
- now the component has a parent
- calls *initialize()* automatically to move to the initialization stage

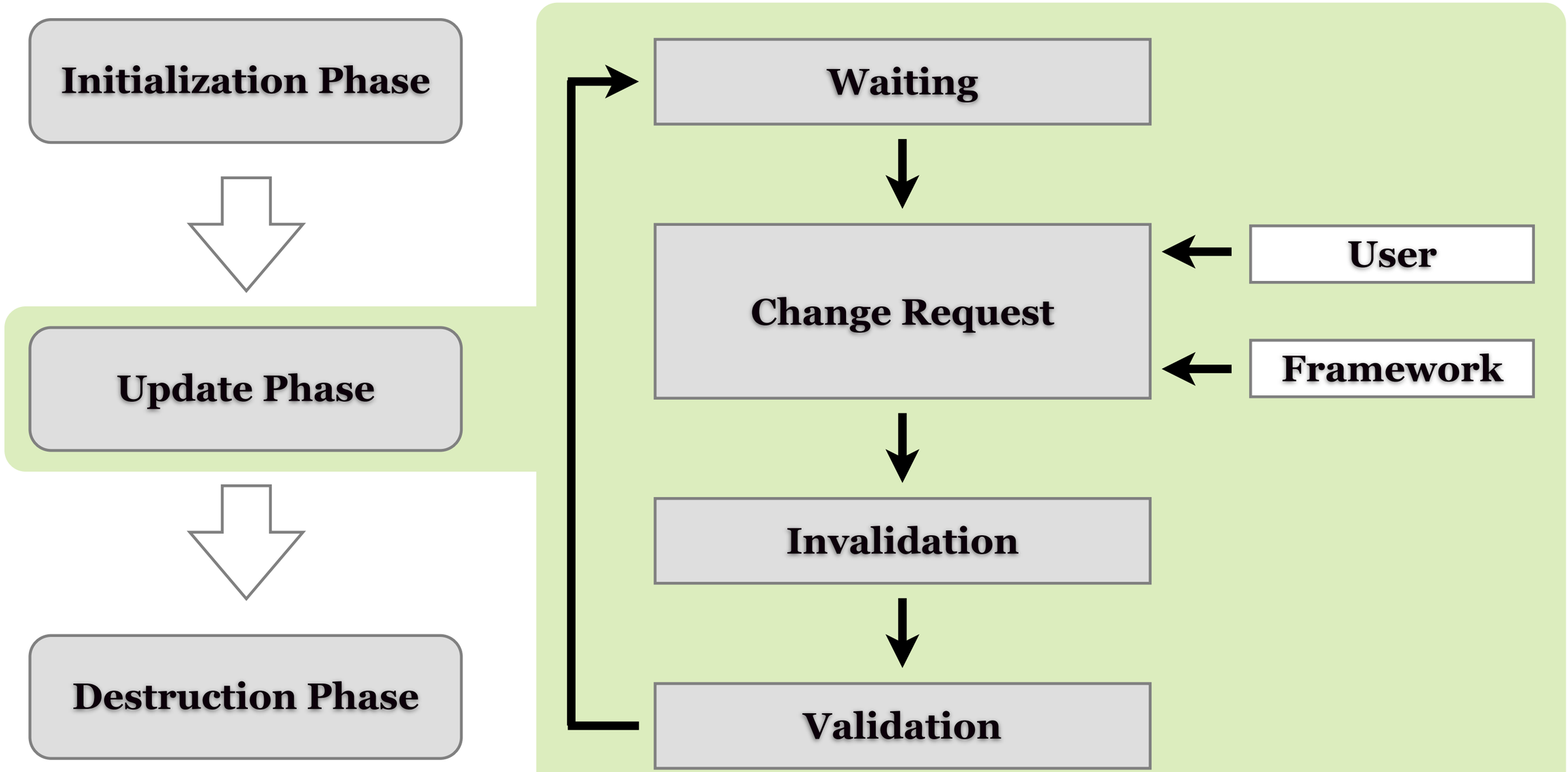


- create children objects
- sizing and positioning
- applying the configured properties and styles

obecto
passionate about software & people



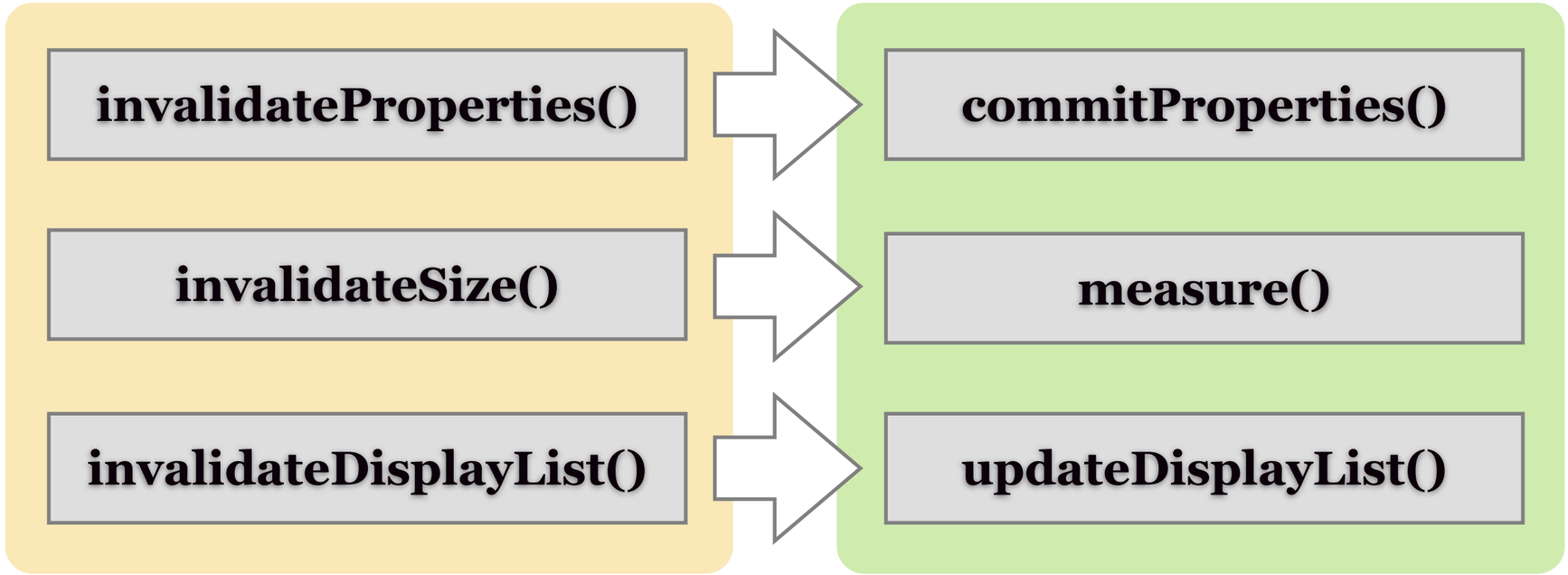
obecto
passionate about software & people



obecto
passionate about software & people

The Invalidation Mechanism

Invalidation Validation



- avoids sequential coupling, or at least such coupling can be located only in the *commitProperties()* method instead of forcing a protocol to the component's clients
- some kind of optimization - prevents unnecessary work in case of setting the property multiple times

invalidateProperties()

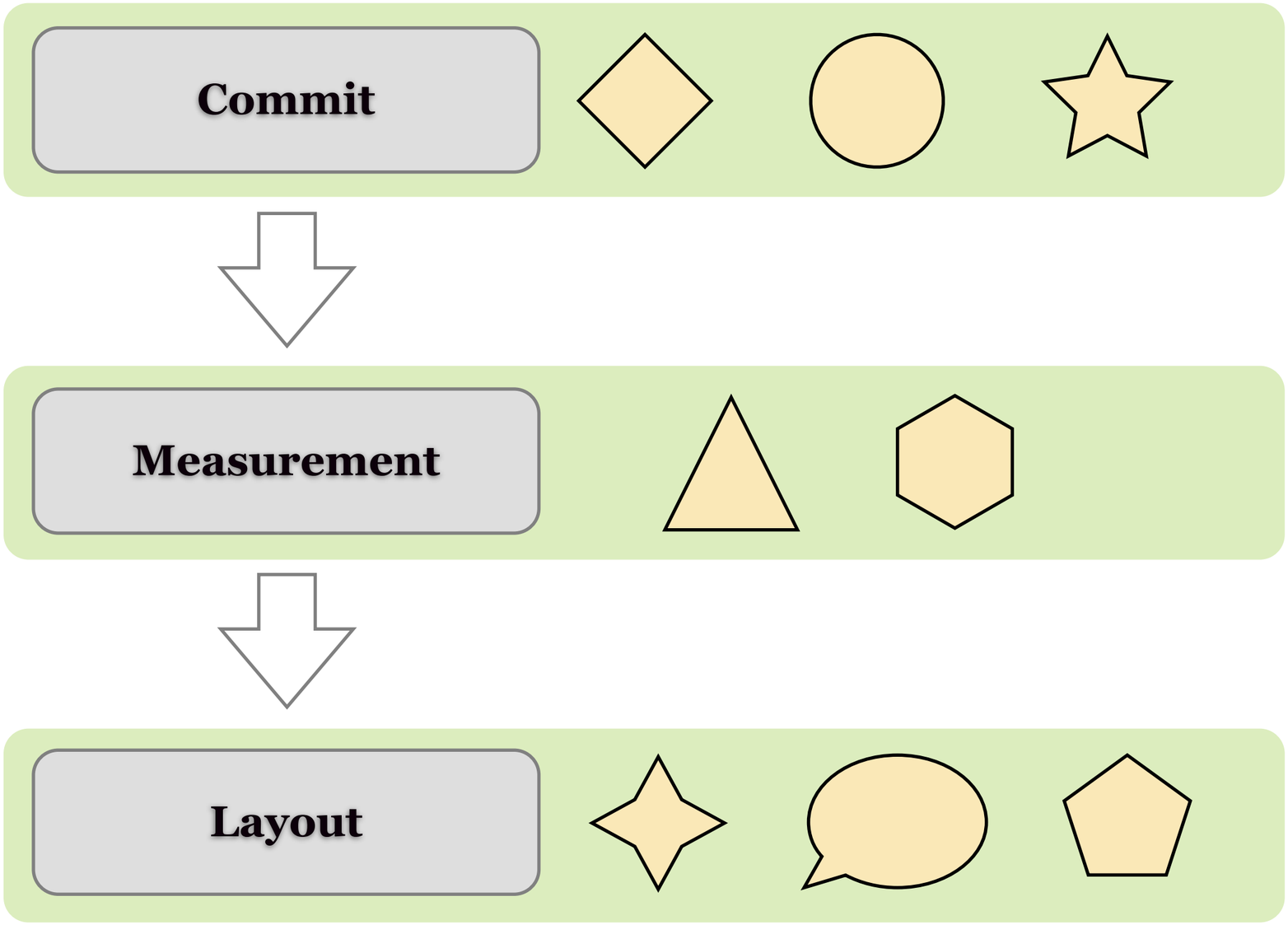
```
public function invalidateProperties():void
{
    if (!invalidatePropertiesFlag)
    {
        invalidatePropertiesFlag = true;

        if (parent && UIComponentGlobals.layoutManager)
            UIComponentGlobals.layoutManager.invalidateProperties(this);
    }
}
```

obecto
passionate about software & people

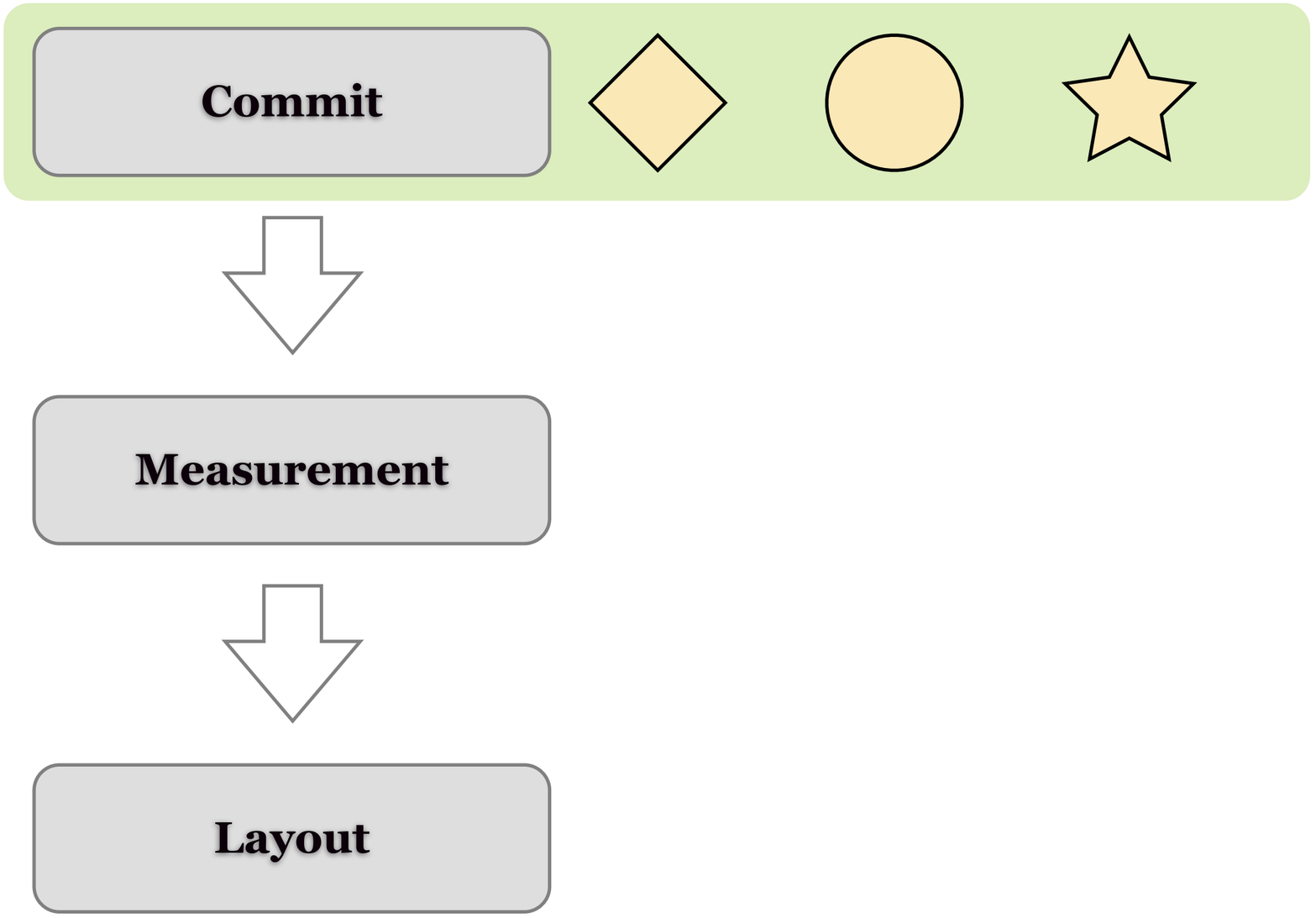
LayoutManager - the engine behind Flex's measurement and layout

obecto
passionate about software & people

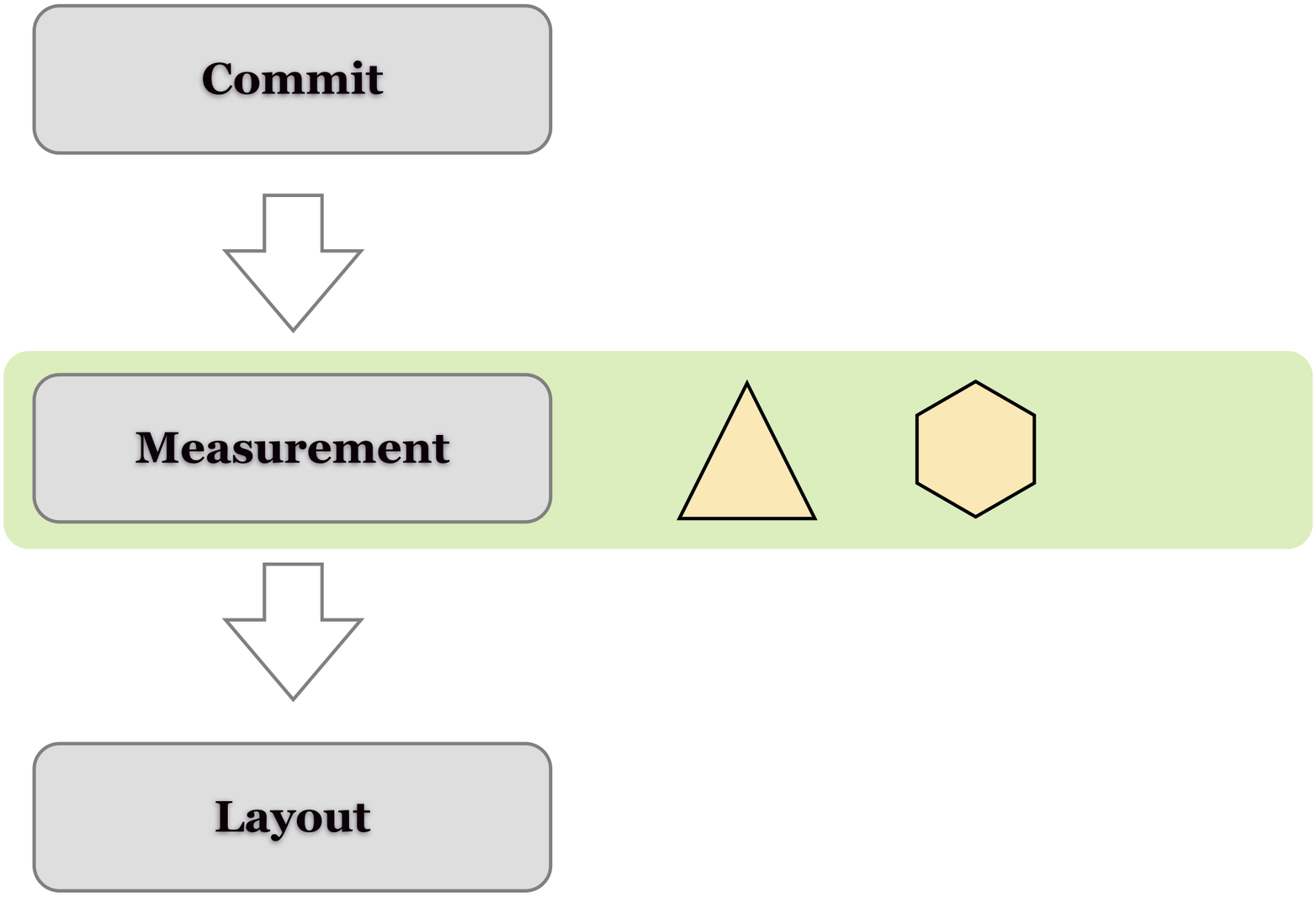
Layout is performed in three phases



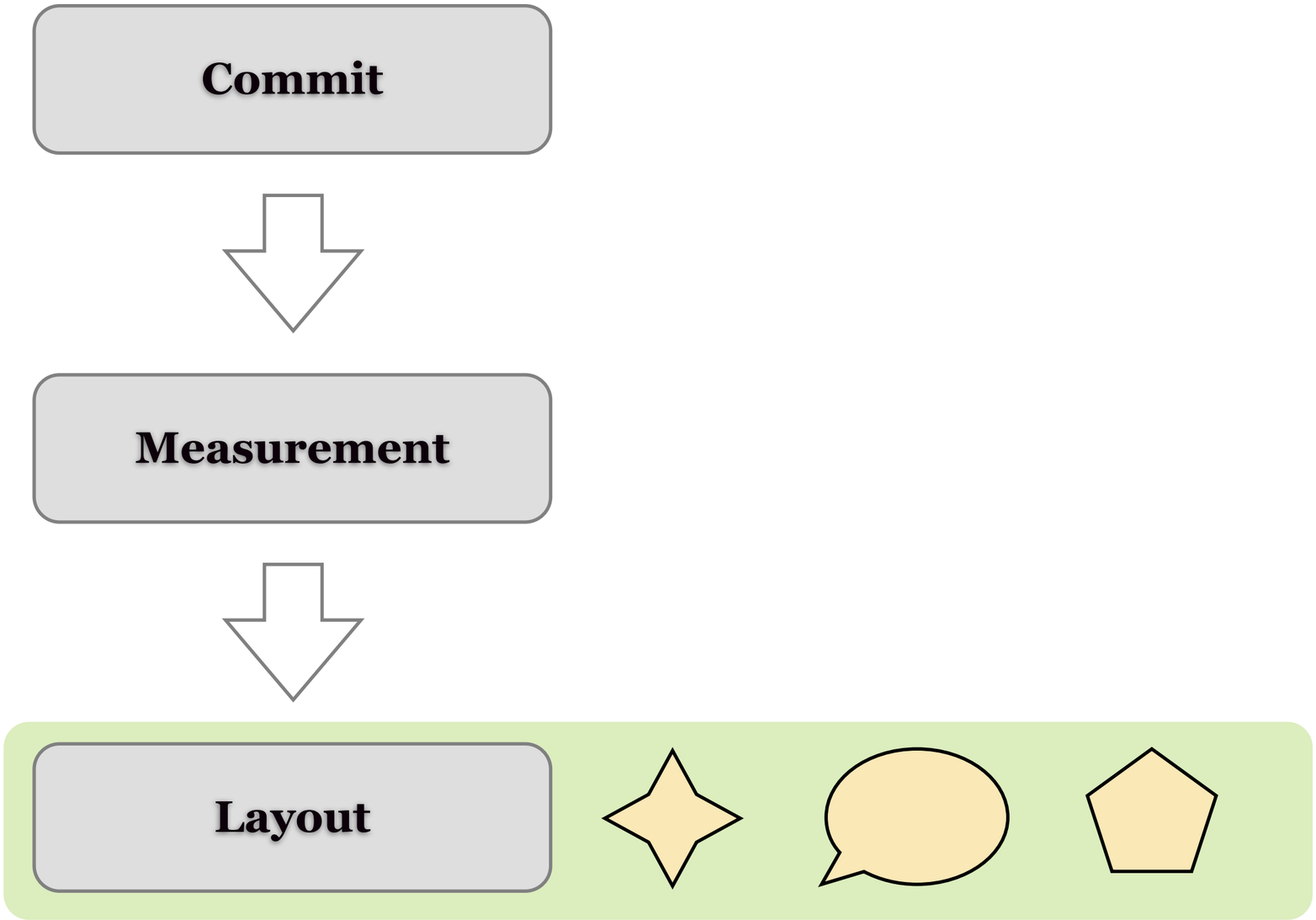
- each phase is processing different UIComponents
- prior to moving to the next phase all components from the current phase are processed
- requests for components to be reprocessed may occur
- such requests are queued for the next run of the phase



- commit begins with a call to *validateProperties*
- *validateProperties* walks through a list of objects and calls their *validateProperties*
- the list is sorted by reversed nesting level (top-down or outside-in ordering)



- begins with a call to *validateSize*
- *validateSize* walks through a list of objects and calls their *validateSize*-method
- the list is sorted by nesting level (starting from the most deeply nested objects)
- if the size or position have changed the object is queued for the layout phase
- additionally, the object's parent is marked for measurement and layout



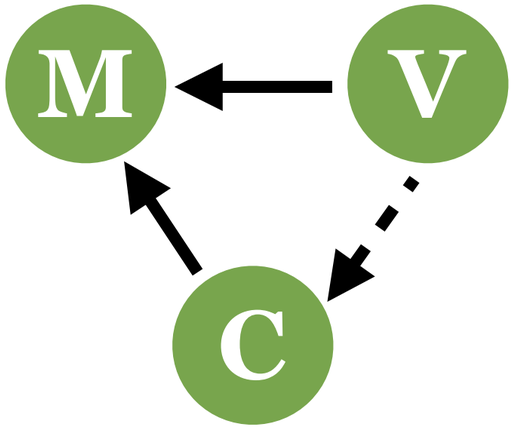
- starts with a call to *validateDisplayList*
- now the list of objects is sorted by reversed nesting order

usePhasedInstantiation

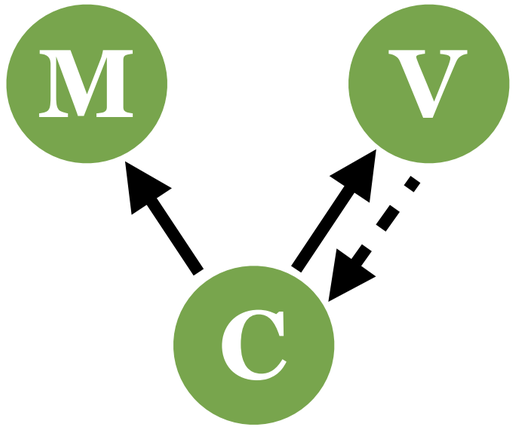
- *LayoutManager.usePhasedInstantiation* is a flag that indicates whether the LayoutManager allows screen updates between phases
- if *true* measurement and layout are done in phases, one phase per screen update
- if *false* all three phases are completed before the screen is updated

MVC Again...

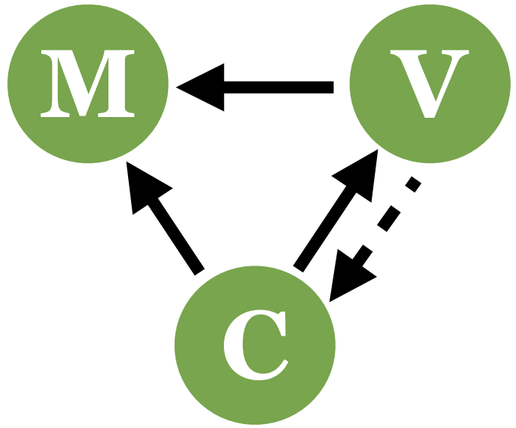
All these are MVCs



view is autonomous, controller only updates the model



the controller is responsible to update the view



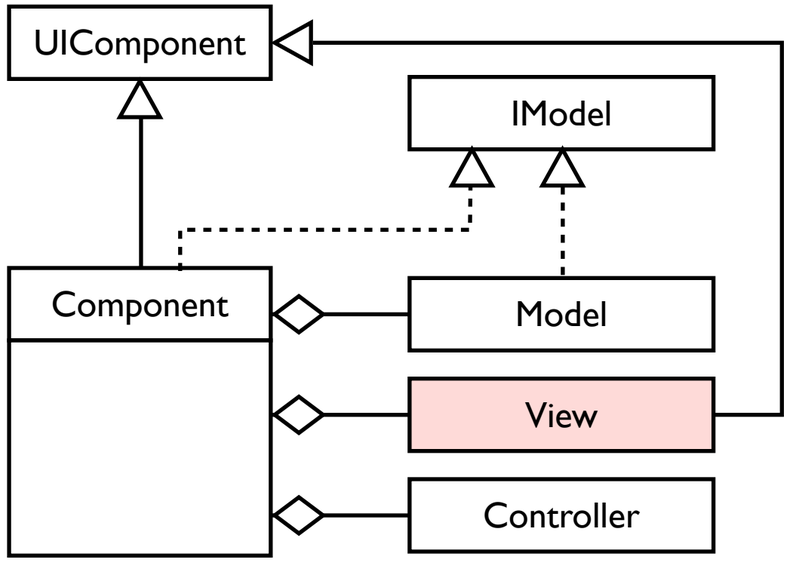
the controller can also command the view

obecto
passionate about software & people

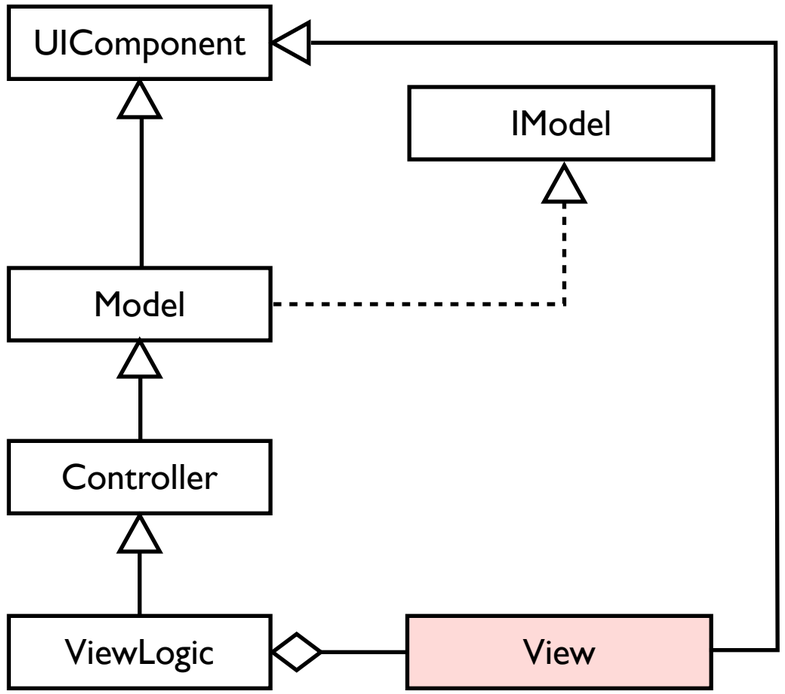
ways to build an MVC...

obecto
passionate about software & people

Composition

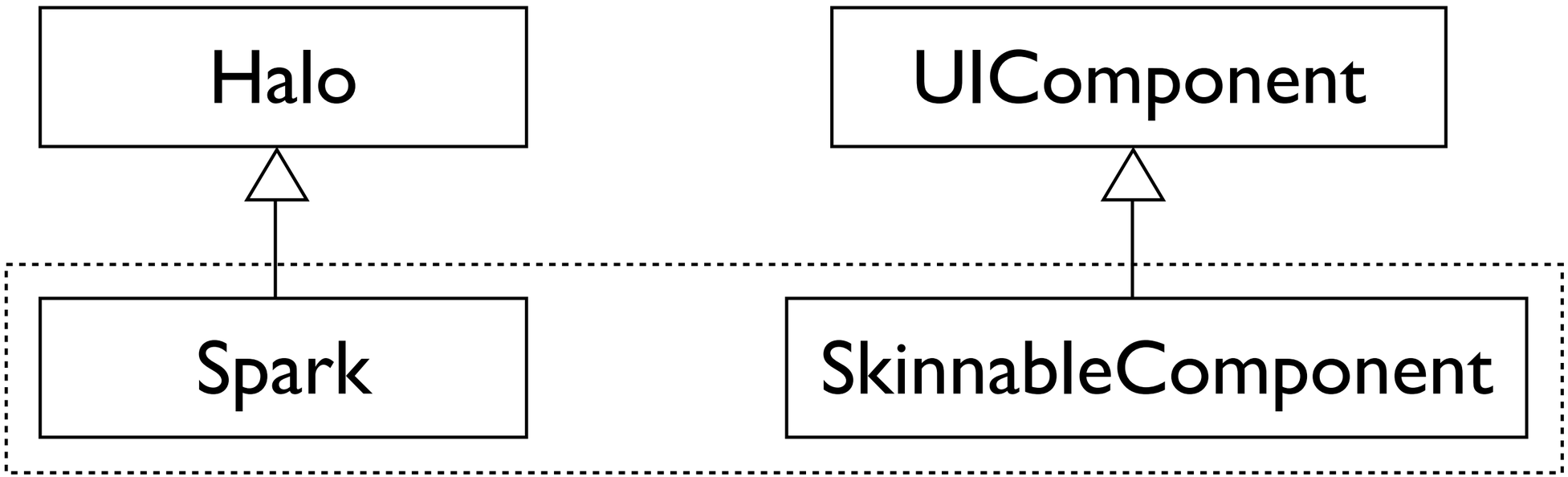


Inheritance



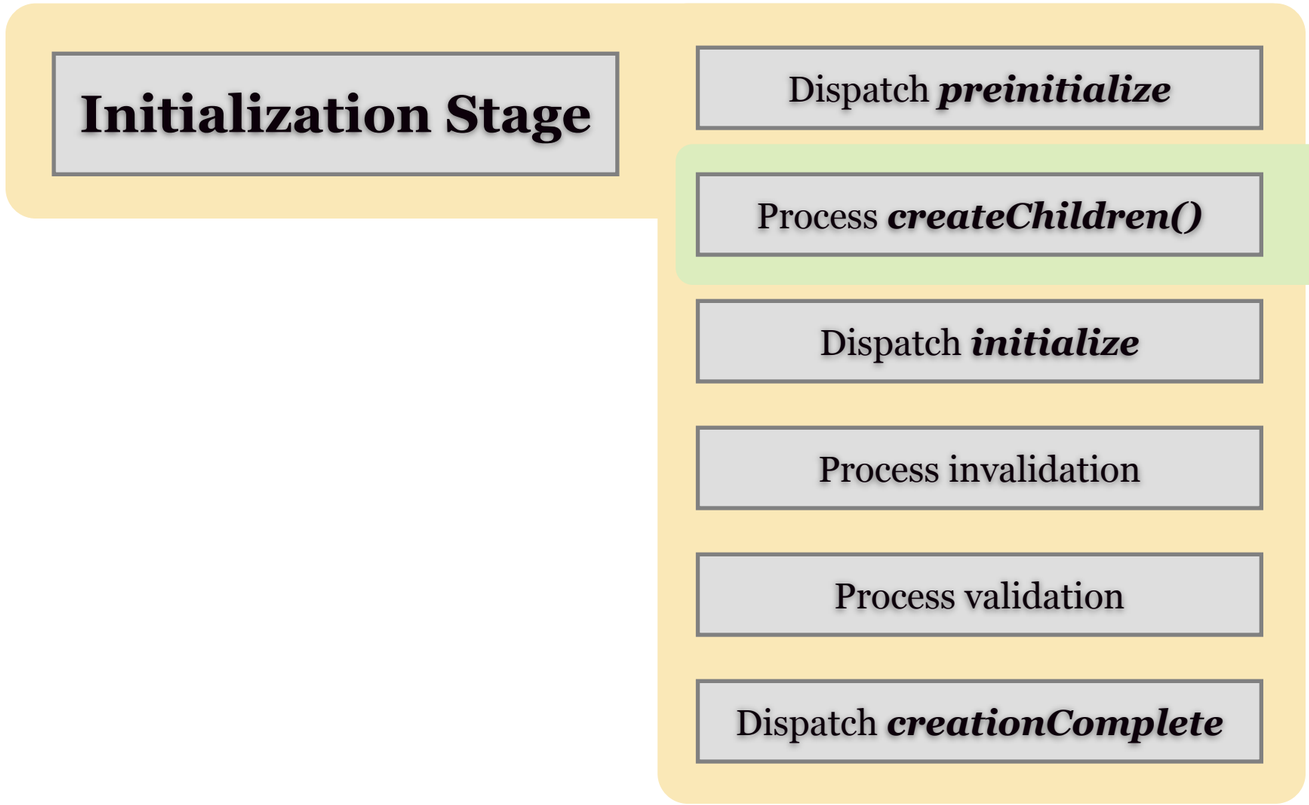
obecto
passionate about software & people

What is Spark?



SkinnableComponent has a bit extended life-cycle...

obecto
passionate about software & people



- *findSkinParts()* would ensure the *designer-developer contract* is fulfilled
- *partAdded()* is executed for each part from the contract
- override *partAdded()* to perform additional initialization (e.g. add listeners)

Let's Build a Spark MVC Component

obecto
passionate about software & people

Model

```
package component.support.periodSelector
{

import spark...SkinnableComponent;

public class TimeRange
      extends SkinnableComponent
{
      [Bindable] public var minDate:Date;
      [Bindable] public var maxDate:Date;
      [Bindable] public var startDate:Date;
      [Bindable] public var endDate:Date;

}

}
```

Controller

```
package component
{
import component.support.periodSelector.Thumb;
import component.support.periodSelector.TimeRange;
import component.support.periodSelector.Timeline;
import spark.components.Button;

[SkinState("normal")]
[SkinState("startDatePopup")]
[SkinState("endDatePopup")]
public class TimePeriodSelector extends TimeRange
{
      [SkinPart(required="true")]
      public var timeline:Timeline;

      [SkinPart(required="true")]
      public var thumb:Thumb;

      [SkinPart(required="true")]
      public var startDateButton:Button;

      [SkinPart(required="true")]
      public var endDateButton:Button;

}

}
```

View (a.k.a. Skin)

```
<?xml version="1.0" encoding="utf-8"?>
<s:Skin xmlns:fx="http://ns.adobe.com/mxml/2009"
     xmlns:s="library://ns.adobe.com/flex/spark"
     xmlns:mx="library://ns.adobe.com/flex/mx"
     xmlns:periodSelector="...periodSelector.*">

     <fx:Metadata>
          [HostComponent("co...TimePeriodSelector")]
     </fx:Metadata>

     <s:states>
          <s:State name="normal" />
          <s:State name="startDatePopup" />
          <s:State name="endDatePopup" />
     </s:states>

     <!-- SkinParts -->
     <s:Button id="startDateButton"/>
     <s:Button id="endDateButton"/>
     <periodSelector:Timeline id="timeline"/>
     <periodSelector:Thumb id="thumb"/>
</s:Skin>
```

obecto
passionate about software & people

Summary

- the are many ways to create and extend Flex Components
- the UIComponent life-cycle
- the invalidating mechanism
- the LayoutManager phases
- different MVCs
- what is Spark?
- the SkinnableComponent life-cycle
- sample Spark MVC Component

obecto
passionate about software & people