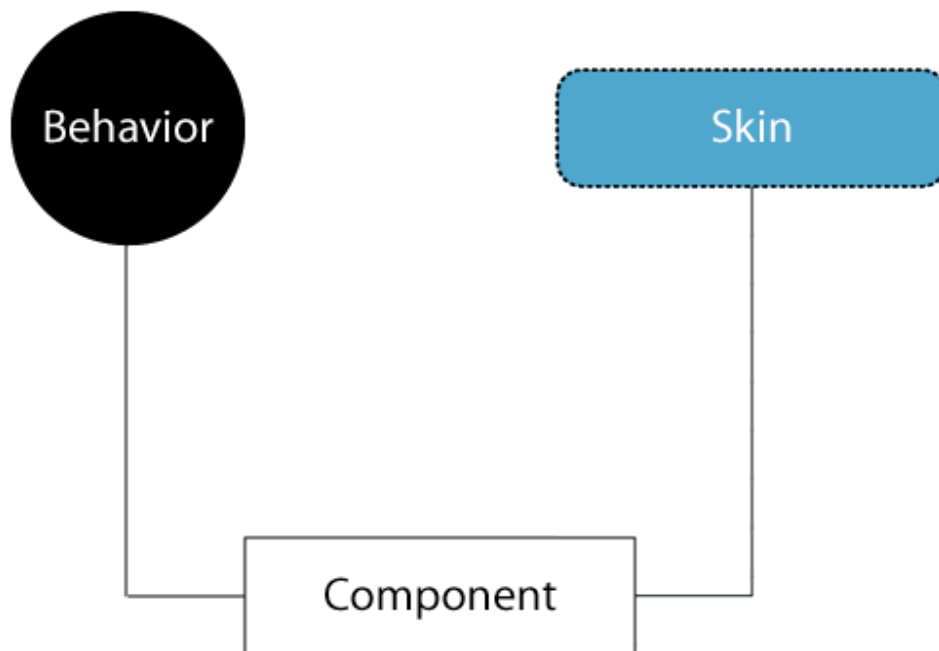
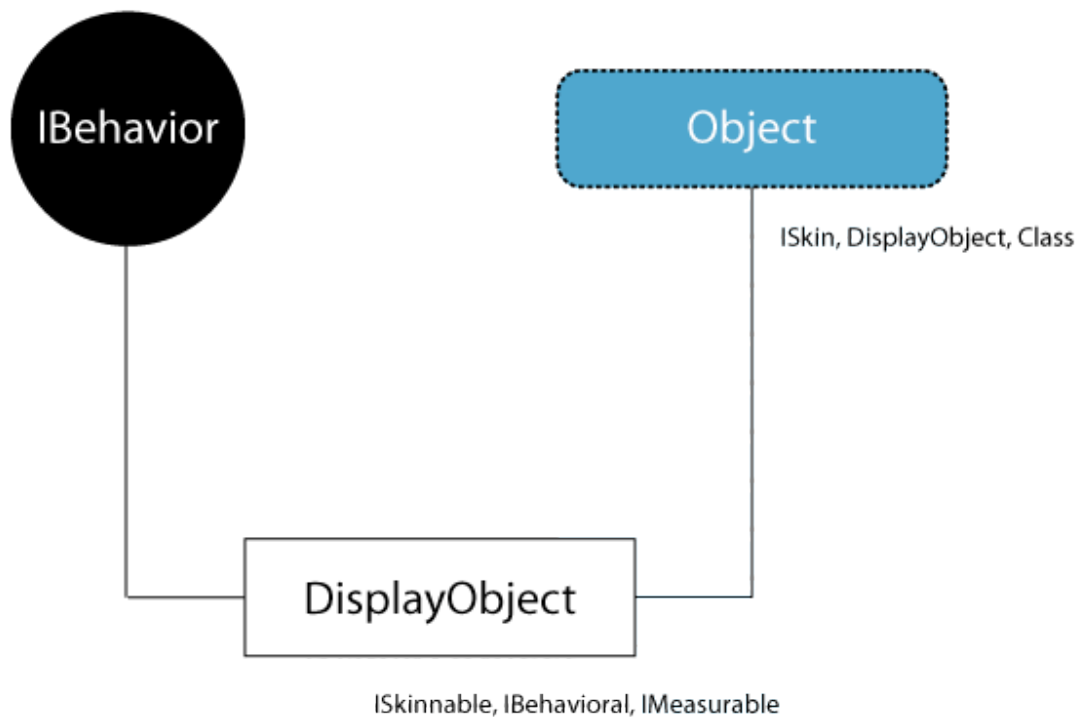


## The Reflex Core Component Model (Draft)

The Reflex Core Component Model is primarily composed of three parts: a Component, Skin, and Behavior, and through these parts a component's API, visual appearance, or interaction can be modified independently without significant risk of regression or code duplication. In this model a Component provides the API (properties, methods, and metadata) used to manage relevant data, a Skin provides the visual definition for a component, and a Behavior controls how a component's API changes in response to user interaction. For convenience, base classes are provided for these three pieces: *reflex.core.Component*, *reflex.skins.Skin*, and *reflex.behaviors.Behavior* respectively.



Although most developers will use the provided base classes for component development, Reflex does not include any expectations that components use these classes or follow this component model. For this reason there is no master “UIComponent” or “UIComponent” interface. Instead, a number of interfaces may be (optionally) composited to define a given component. These interfaces include *ISkinnable*, *IBehavioral*, and *IMeasurable* to define components, as well as *ISkin* and *IBehavior* to define a Skin or Behavior respectively.



Reflex only requires that components extend *DisplayObject* for use as children or for participation in layouts etc. Similarly components which implement *ISkinnable* only require that the assigned skin extend *Object*. In this way *DisplayObjects* such as *MovieClips*, *Classes* such as embedded image assets, or *ISkin* definitions (which don't extend *DisplayObject*) may be used. Components which implement *IBehavioral* should expect that an assigned behavior implement *IBehavior*, but complex behaviors may be composited using a *CompositeBehavior* instance.

When following this component model Skins and Behaviors are easily replaced and reused by similar Components, and new component classes are created only by defining the properties and methods required to manage data. This separation of the API from both it's visual appearance and interaction makes it possible to reuse the majority of component code in both Flash-only and Flex application environments. Reflex can, then, provide light-weight components which are available to all Flash environments by creating a base Component class which is conditionally (or manually) compiled to extend UIComponent (for Flex) or MovieClip (for other use cases).

