

Mission: Impossible

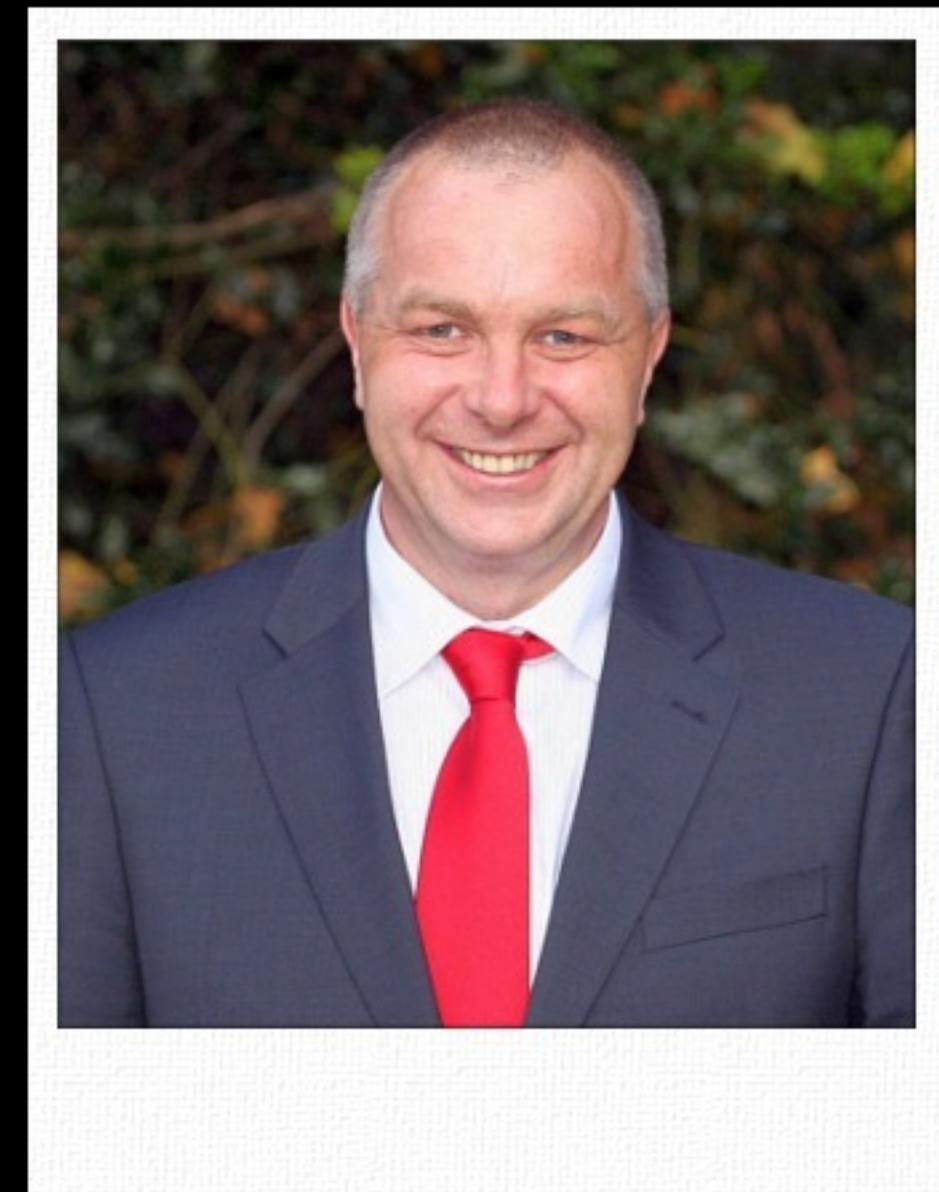
Purely declarative User Interface Modeling

Achim Demelt, Wilken GmbH

Part I: Introduction

The Challenge

"We want to write the complete UI of our next generation ERP System in a declarative way. Not a single imperative statement."



Volker Mailach
May 2011

The Context



S4 



The Motivation




The Goals



Declarative
Style



Visual
Conformity



Extensibility

Part II: The Mission

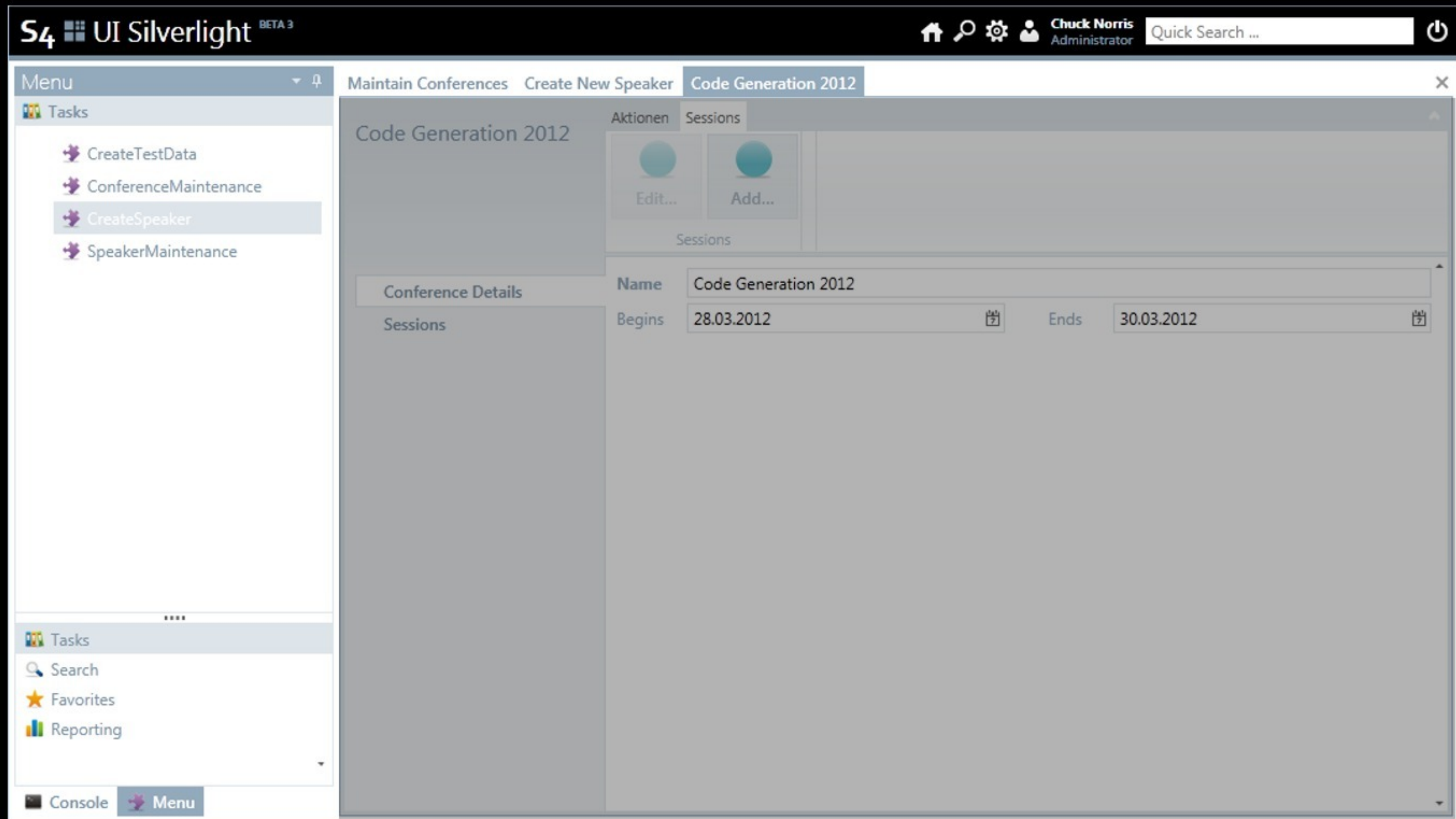
The Core Abstractions

The screenshot displays the S4 UI Silverlight application interface. The top navigation bar includes the S4 logo, the text 'UI Silverlight BETA 3', and user information for 'Chuck Norris Administrator' with a search bar. The left sidebar contains a 'Menu' section with 'Tasks' (CreateTestData, ConferenceMaintenance, CreateSpeaker, SpeakerMaintenance) and a bottom section with 'Search', 'Favorites', and 'Reporting'. The main content area is titled 'Code Generation 2012' and features tabs for 'Aktionen' and 'Sessions'. The 'Sessions' tab is active, showing a table with the following data:

Name	Begins	Ends
Code Generation 2012	28.03.2012	30.03.2012

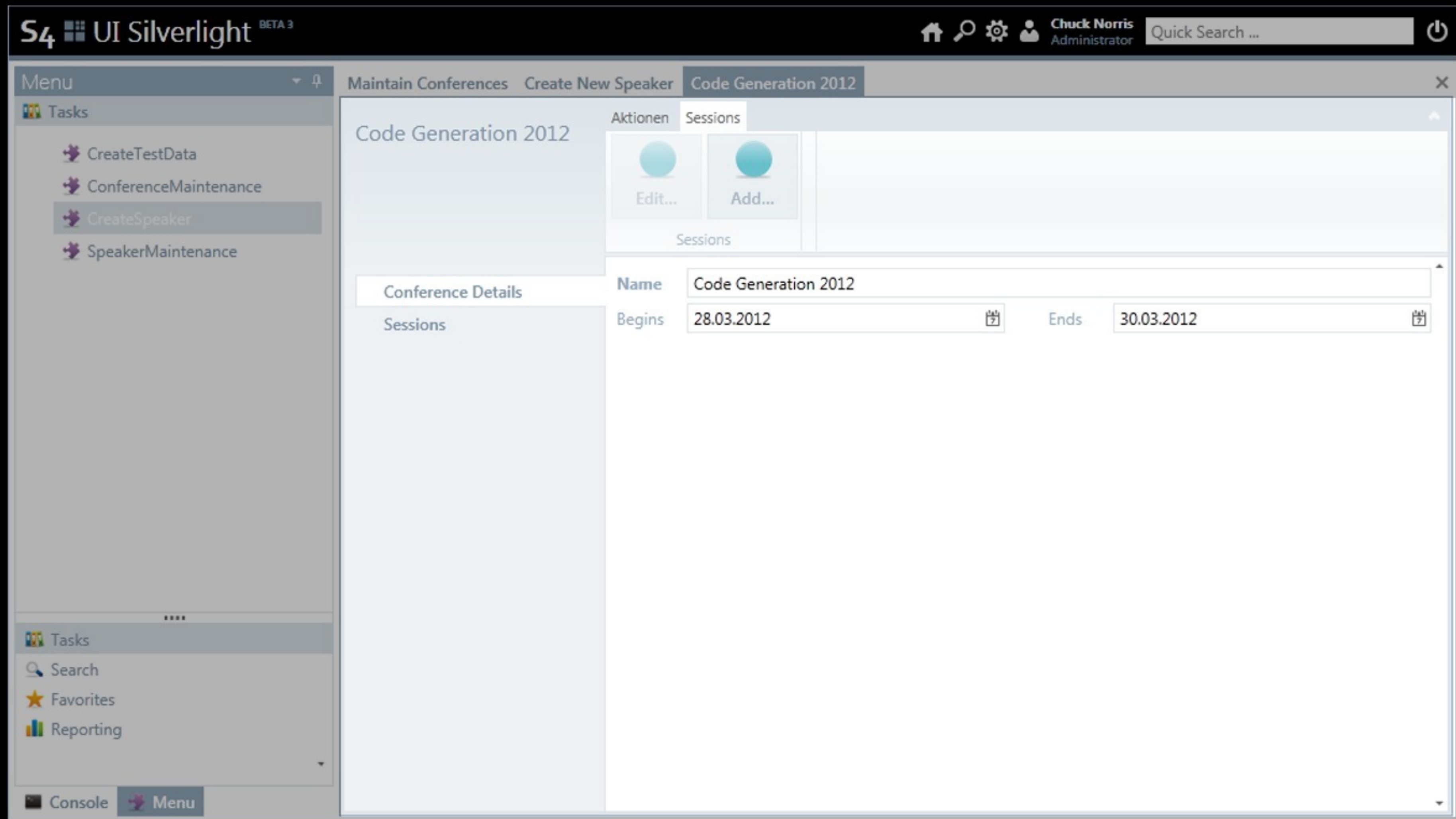
The interface also includes a 'Console' and 'Menu' button at the bottom left.

The Core Abstractions



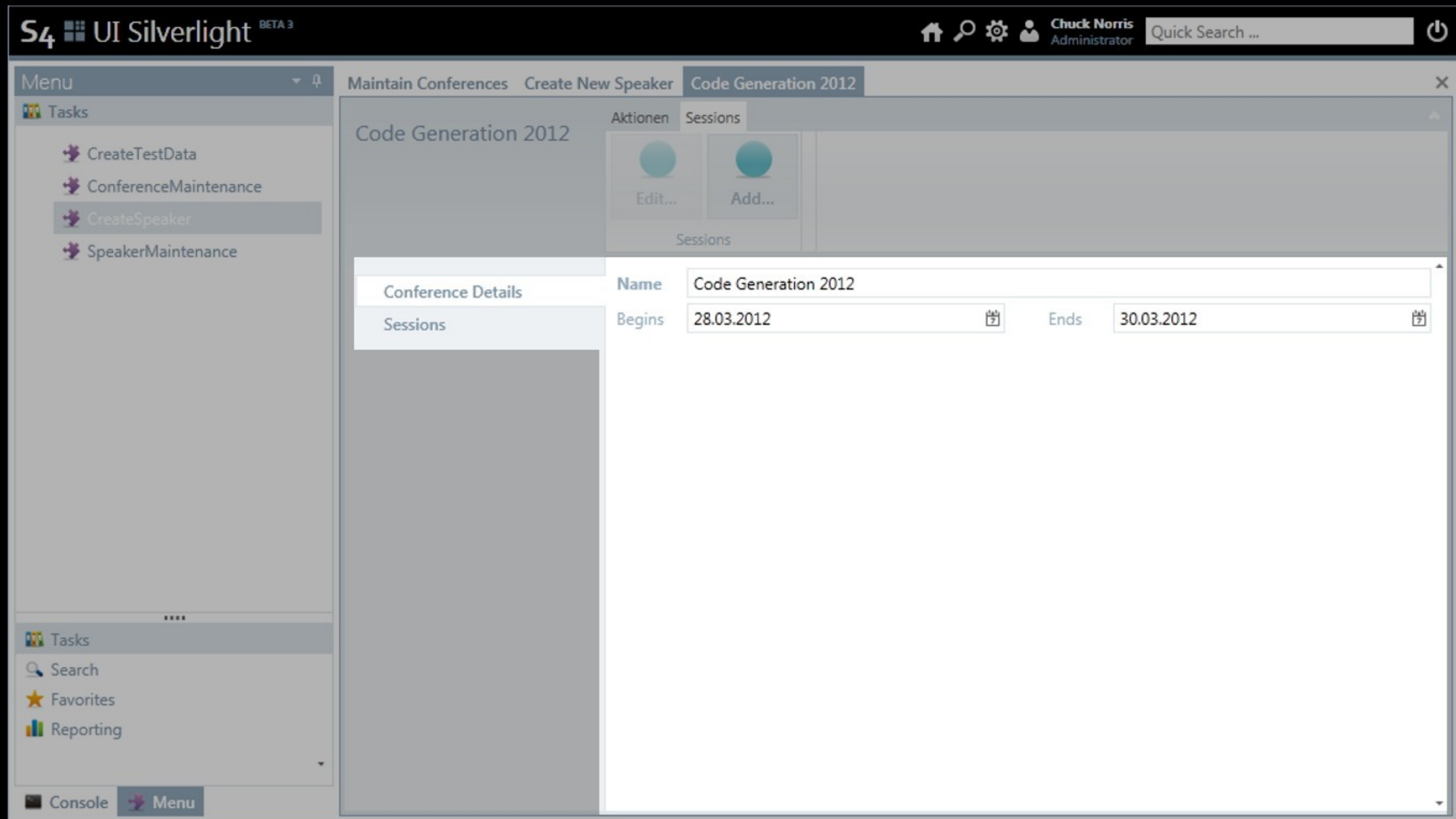
Application Frame

The Core Abstractions



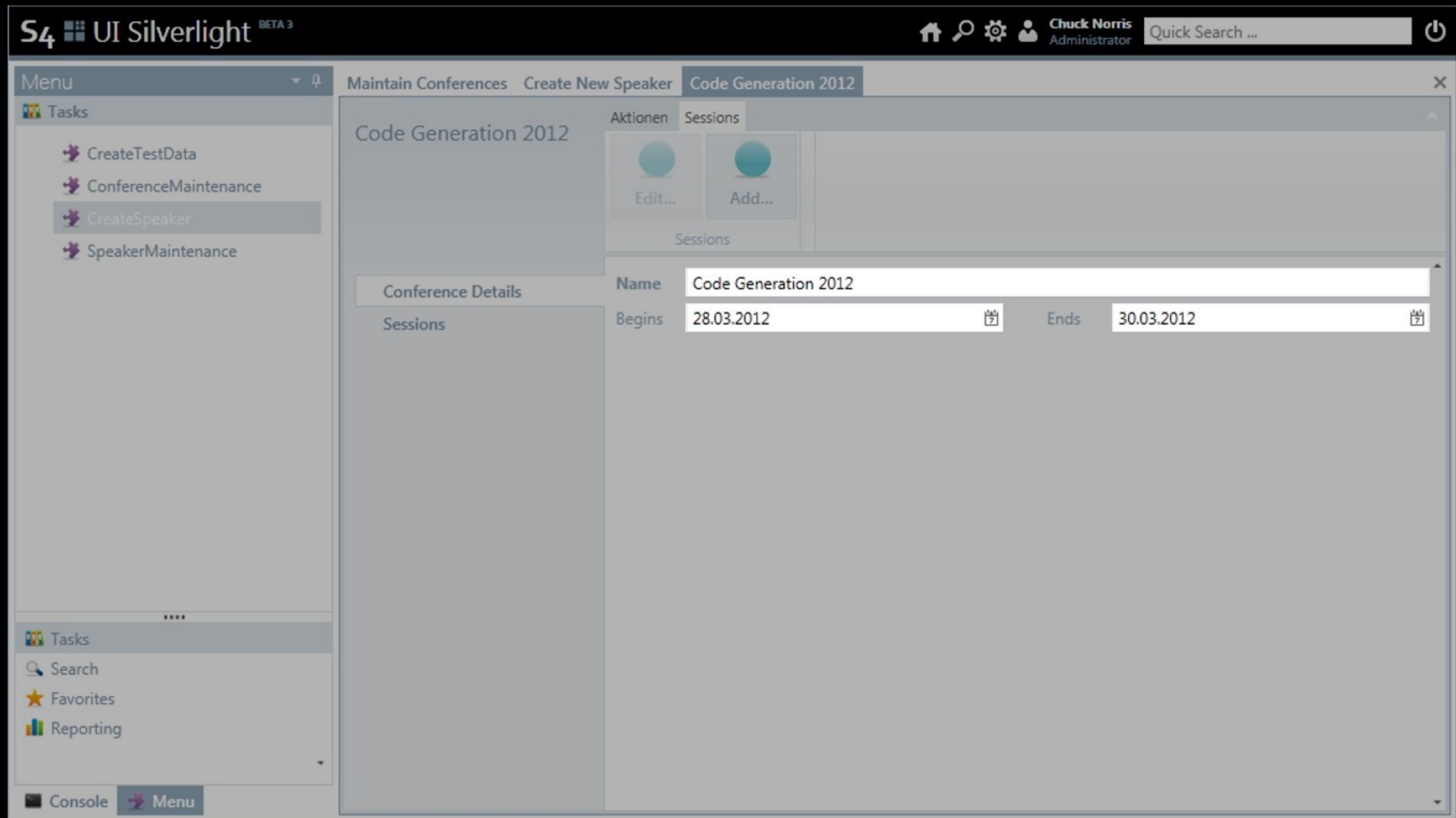
Task

The Core Abstractions



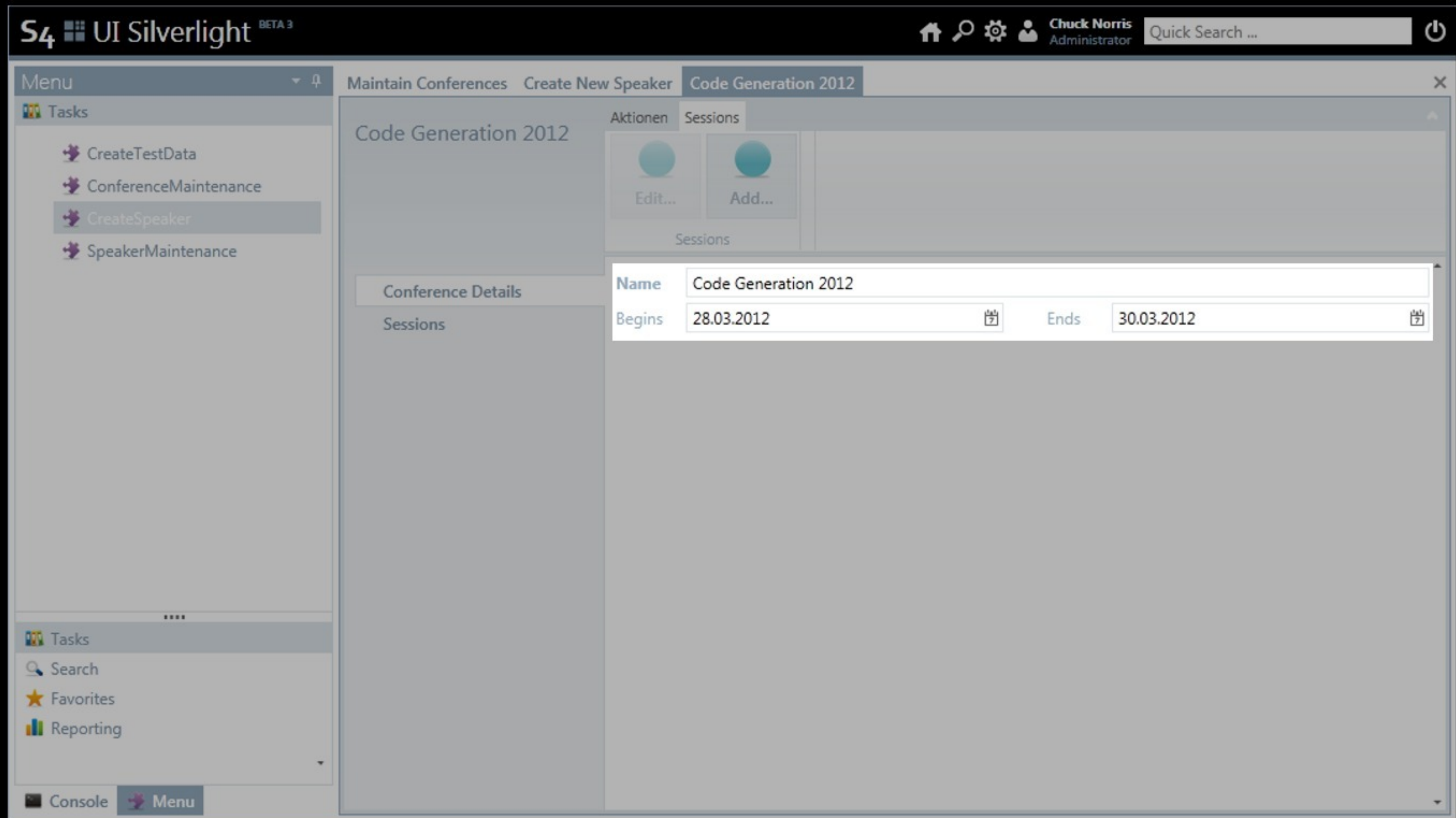
Section

The Core Abstractions



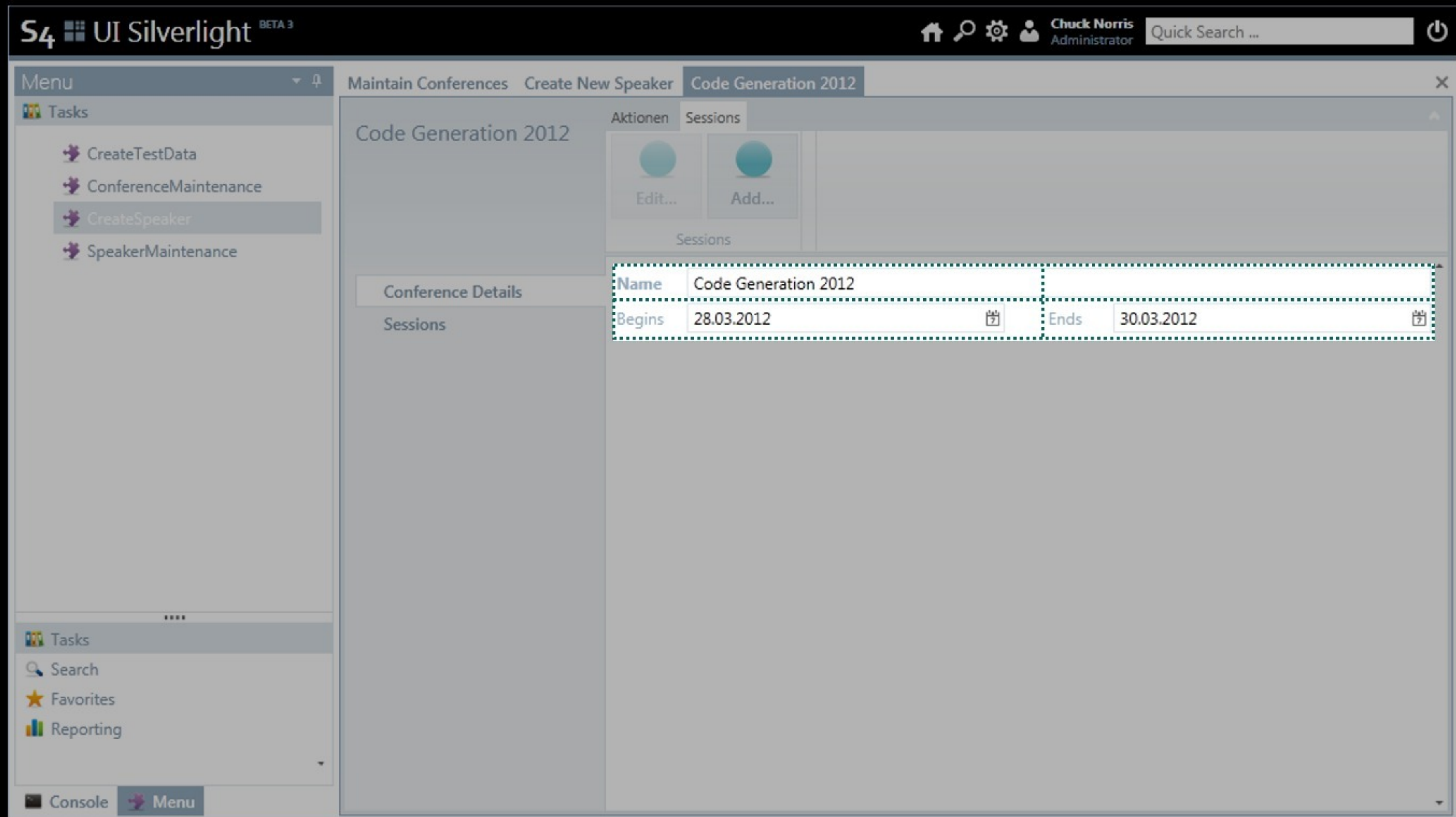
UiComponent

The Core Abstractions



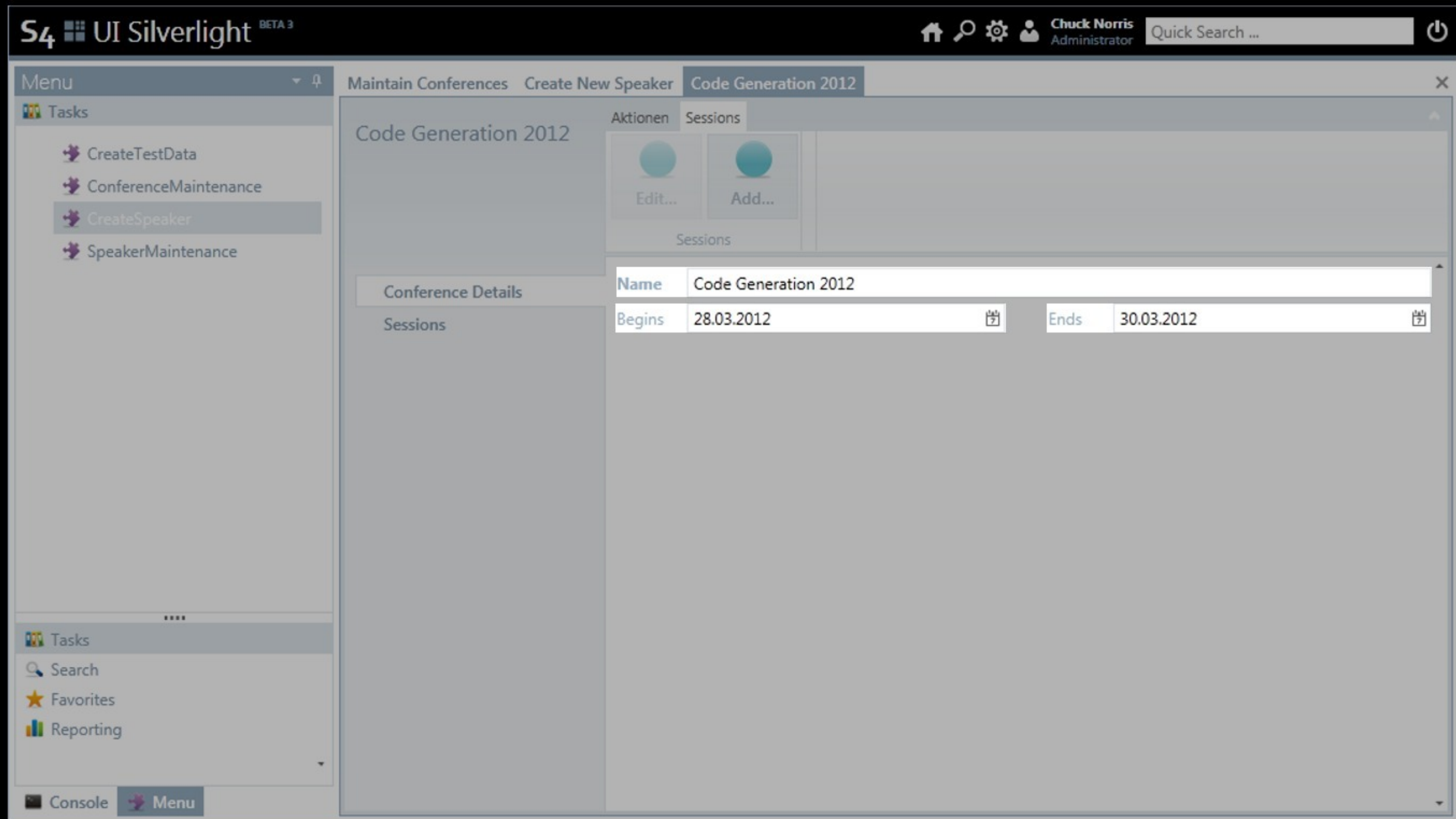
UiComponent

The Core Abstractions



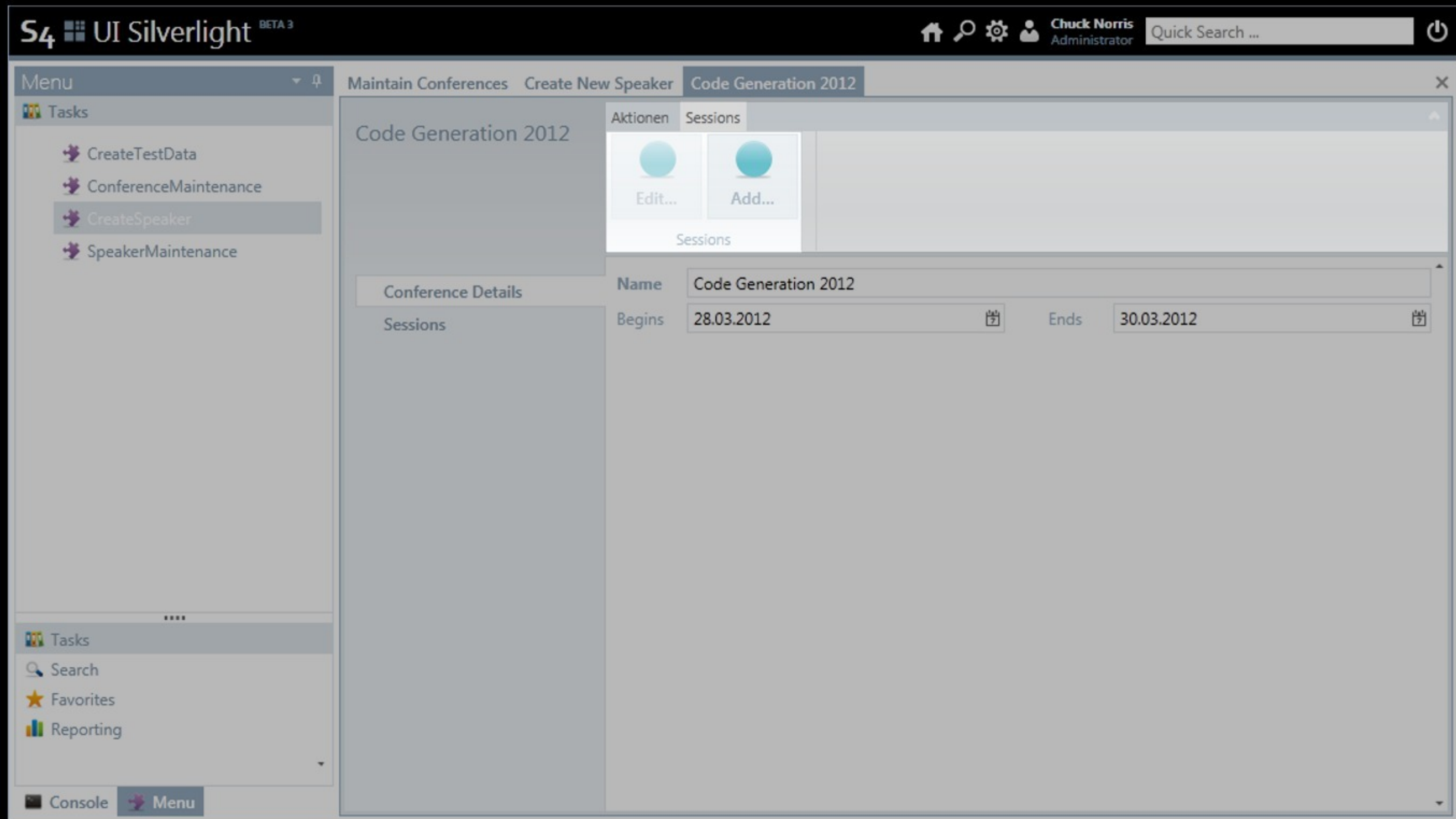
Layout

The Core Abstractions



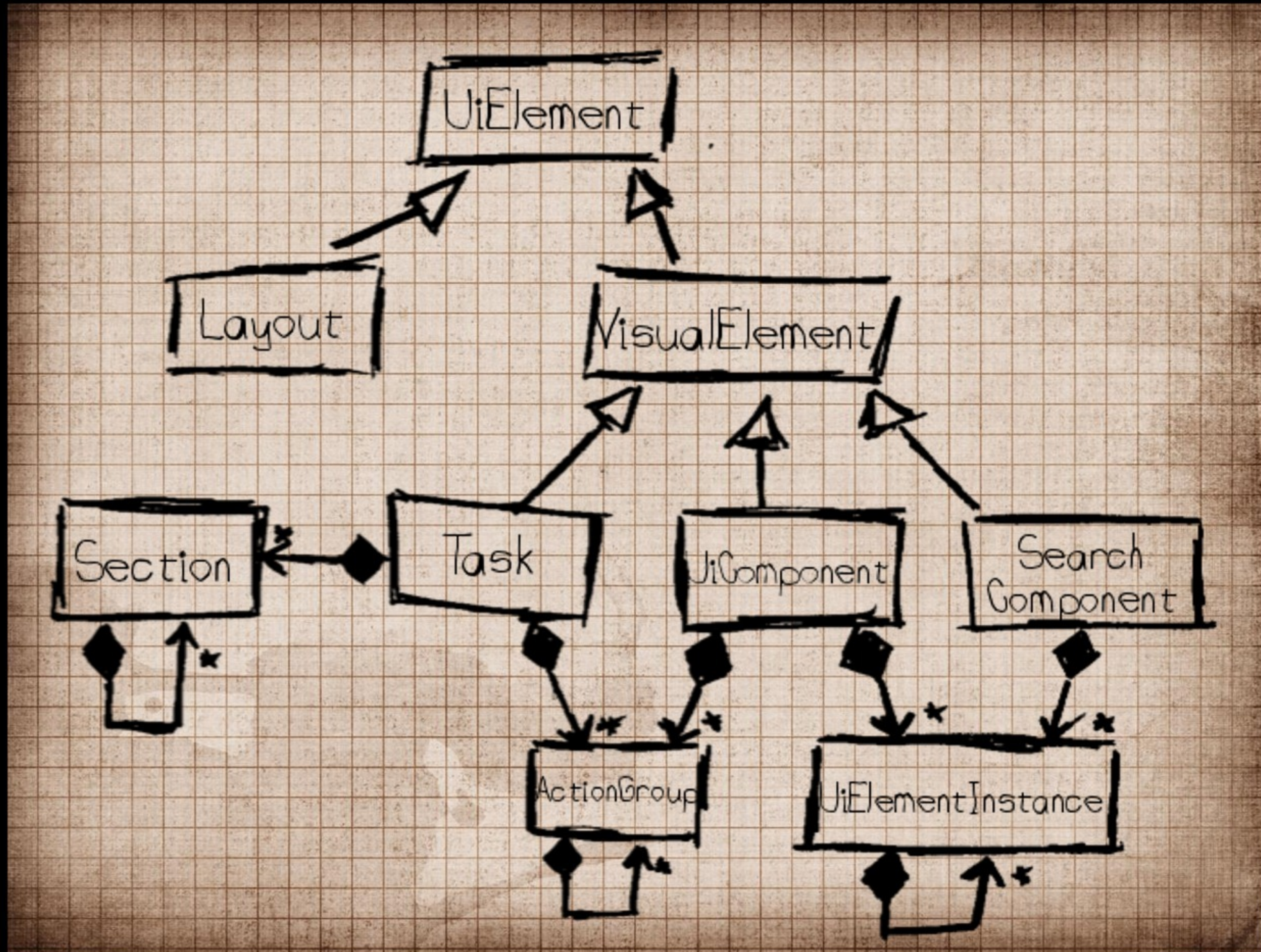
UiElementInstance

The Core Abstractions



ActionGroup

The Metamodel



The Native Components

```
Ui.s4 X
ui System.Ui.Ui;

// layouts
native layout Flow(LayoutOrientation orientation = LayoutOrientation.vertical)
native layout Form(Integer numColumns = 2)
native layout Span(Integer spanColumns = 99)
native layout Gap(Integer gapCount = 1)
+ native layout Tabs(...)

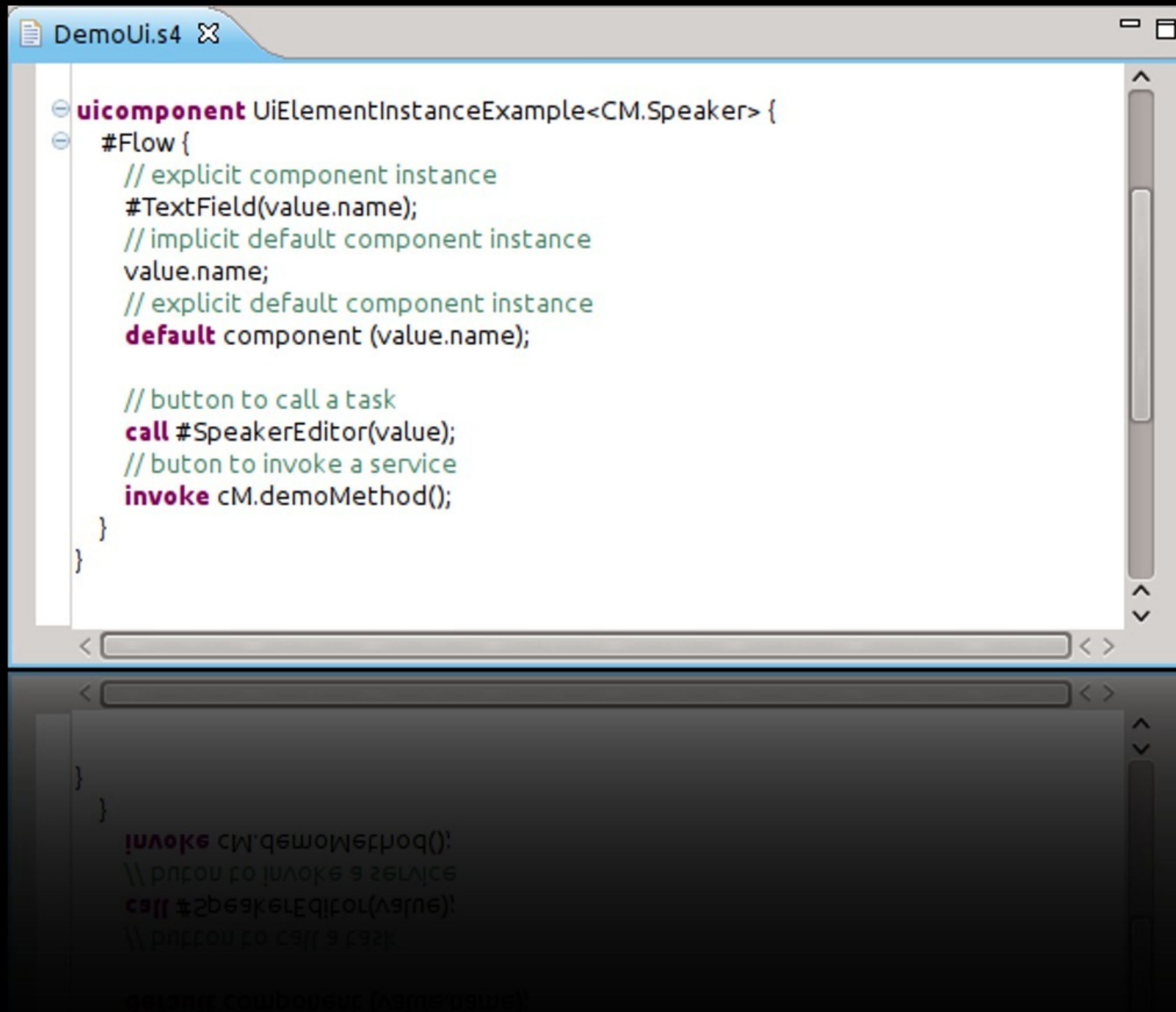
// components
native uicomponent Label<String>

native uicomponent TextField<String>(Integer maxLength=null, String format=null)
native uicomponent TextArea<String>(Integer maxLength=null, Integer lines=7)
native uicomponent PasswordField<String>

native uicomponent ShortField<Short>(Integer digits = null, String format=null)
native uicomponent IntegerField<Integer>(Integer digits = null, String format=null)
native uicomponent LongField<Long>(Integer digits = null, String format=null)
+ native uicomponent DecimalField<Decimal>(...)
```

```
+ native uicomponent DecimalField<Decimal>(...)
native uicomponent RoundField<Round>(Integer digits = null, String format=null)
native uicomponent IntegerField<Integer>(Integer digits = null, String format=null)
native uicomponent ShortField<Short>(Integer digits = null, String format=null)
native uicomponent PasswordField<String>
native uicomponent TextField<String>(Integer maxLength=null, Integer lines=7)
native uicomponent TextField<String>(Integer maxLength=null, String format=null)
```


The UiElementInstance Concept

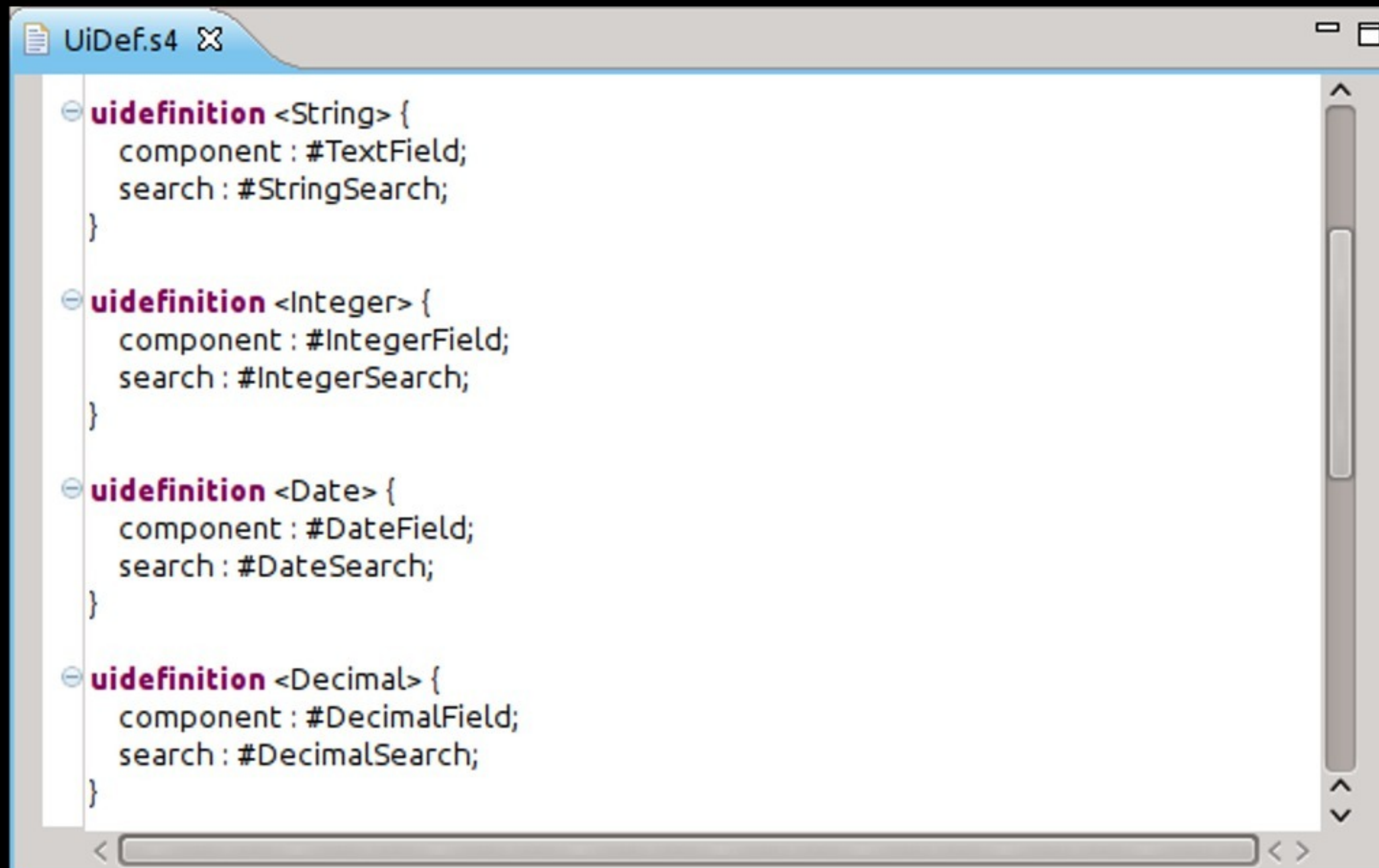


```
DemoUi.s4 X
```

```
uicomponent UiElementInstanceExample<CM.Speaker> {  
  #Flow {  
    // explicit component instance  
    #TextField(value.name);  
    // implicit default component instance  
    value.name;  
    // explicit default component instance  
    default component (value.name);  
  
    // button to call a task  
    call #SpeakerEditor(value);  
    // button to invoke a service  
    invoke CM.demoMethod();  
  }  
}
```

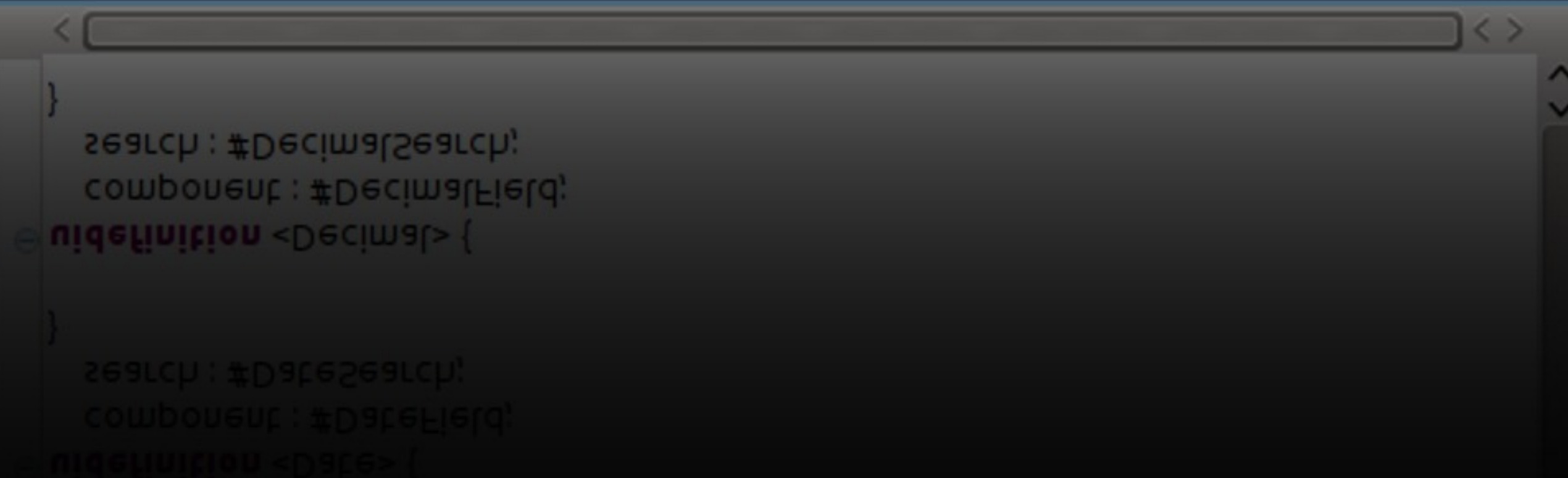
The image shows a code editor window titled 'DemoUi.s4'. The code defines a UI component 'UiElementInstanceExample' that takes a 'CM.Speaker' parameter. Inside a '#Flow' block, it demonstrates three ways to instantiate a component: explicitly with '#TextField(value.name)', implicitly by using the parameter 'value.name' directly, and explicitly with a 'default' component. It also shows how to call a task ('call #SpeakerEditor(value)') and invoke a service ('invoke CM.demoMethod()'). The code is color-coded: keywords like 'uicomponent', '#Flow', 'call', 'invoke', and 'default' are in purple; comments are in green; and identifiers like 'CM.Speaker', 'value.name', and 'CM.demoMethod()' are in black.

The Global Defaults



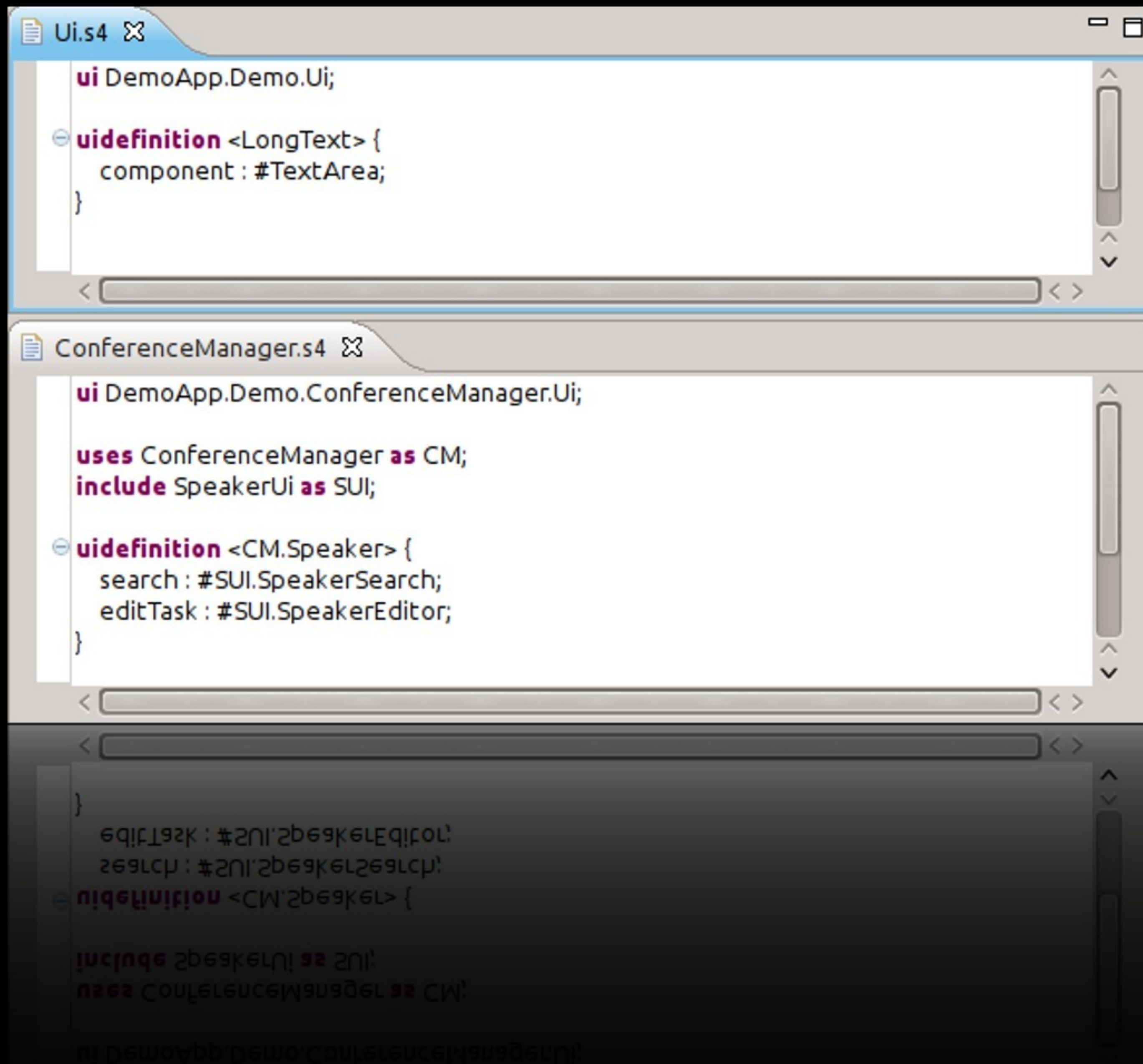
```
UiDef.s4
```

```
udefinition <String> {  
  component : #TextField;  
  search : #StringSearch;  
}  
  
udefinition <Integer> {  
  component : #IntegerField;  
  search : #IntegerSearch;  
}  
  
udefinition <Date> {  
  component : #DateField;  
  search : #DateSearch;  
}  
  
udefinition <Decimal> {  
  component : #DecimalField;  
  search : #DecimalSearch;  
}
```



```
udefinition <Decimal> {  
  search : #DecimalSearch;  
  component : #DecimalField;  
}  
  
udefinition <Date> {  
  search : #DateSearch;  
  component : #DateField;  
}
```


The Application's Defaults



```
Ui.s4
ui DemoApp.Demo.Ui;

uidefinition <LongText> {
  component : #TextArea;
}

ConferenceManager.s4
ui DemoApp.Demo.ConferenceManager.Ui;

uses ConferenceManager as CM;
include SpeakerUi as SUI;

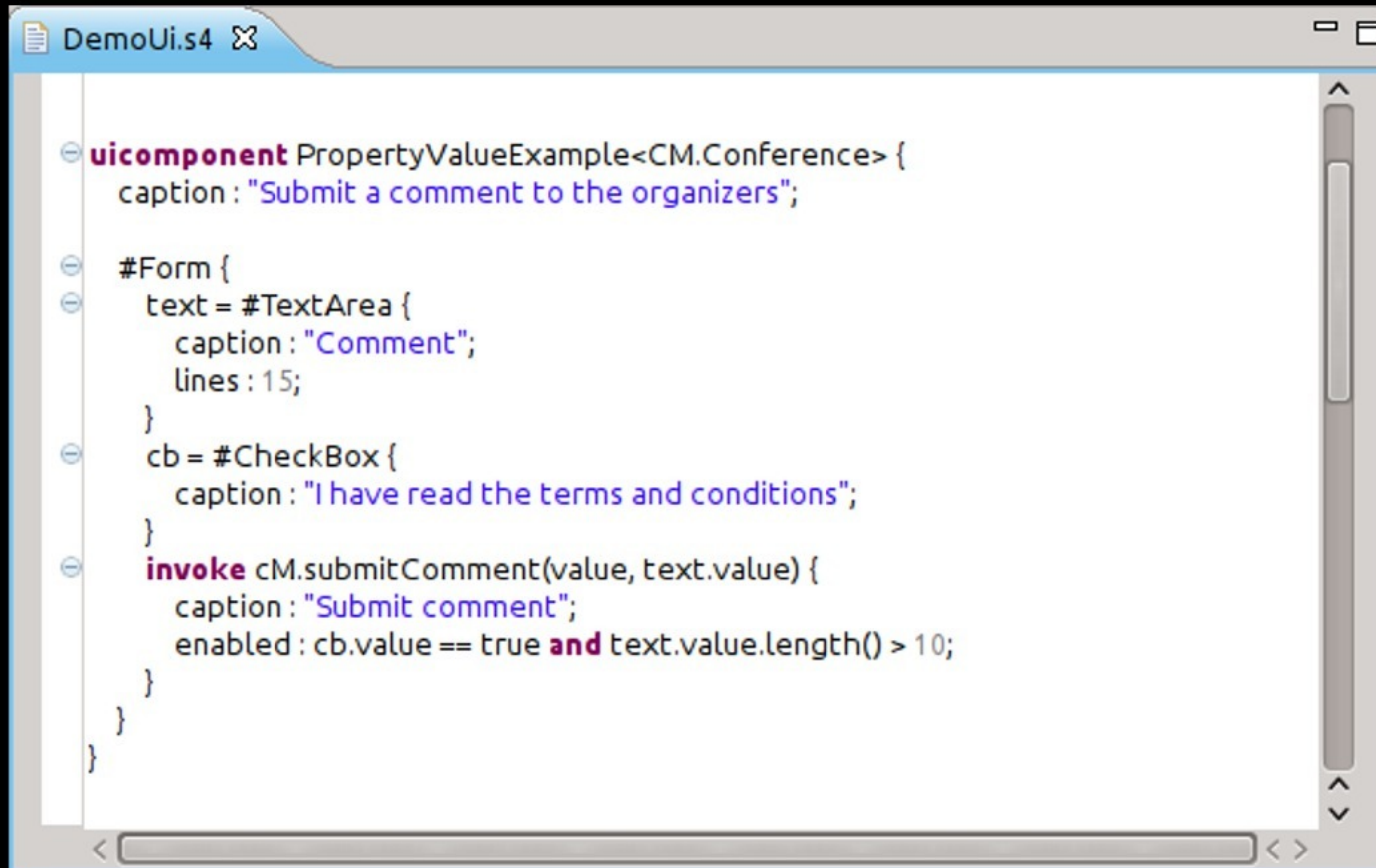
uidefinition <CM.Speaker> {
  search : #SUI.SpeakerSearch;
  editTask : #SUI.SpeakerEditor;
}

}
editTask : #SUI.SpeakerEditor;
search : #SUI.SpeakerSearch;
uidefinition <CM.Speaker> {

include SpeakerUi as SUI;
uses ConferenceManager as CM;

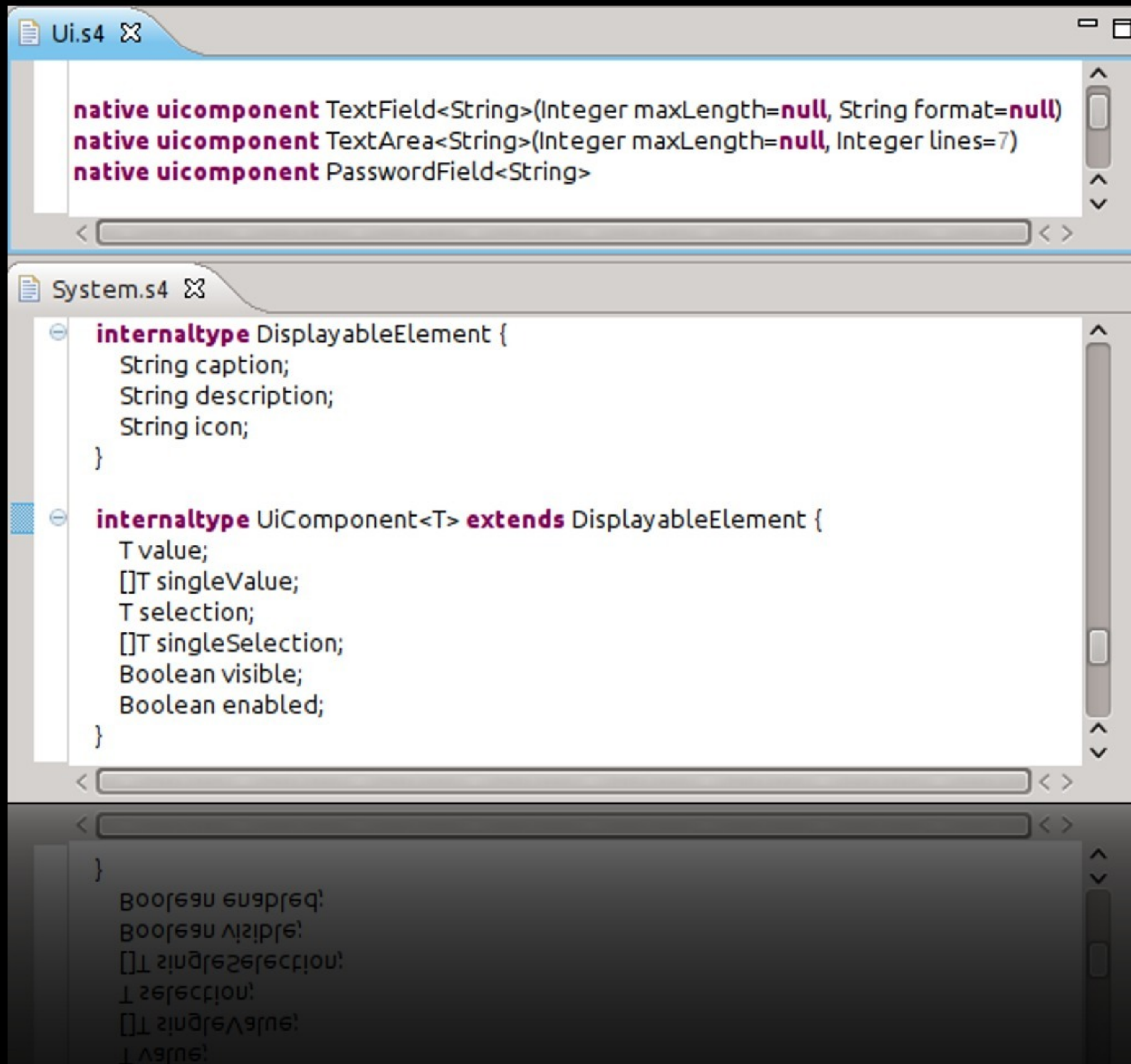
ui DemoApp.Demo.ConferenceManager.Ui;
```

The PropertyValue Concept (1)



```
uicomponent PropertyValueExample<CM.Conference> {  
  caption : "Submit a comment to the organizers";  
  
  #Form {  
    text = #TextArea {  
      caption : "Comment";  
      lines : 15;  
    }  
    cb = #CheckBox {  
      caption : "I have read the terms and conditions";  
    }  
    invoke cM.submitComment(value, text.value) {  
      caption : "Submit comment";  
      enabled : cb.value == true and text.value.length() > 10;  
    }  
  }  
}
```


The PropertyValue Concept (2)



The image shows a code editor with two tabs: `Ui.s4` and `System.s4`. The `Ui.s4` tab is active and shows three native ui component definitions. The `System.s4` tab is also visible and shows two internal type definitions.

```
Ui.s4
native ui component TextField<String>(Integer maxLength=null, String format=null)
native ui component TextArea<String>(Integer maxLength=null, Integer lines=7)
native ui component PasswordField<String>

System.s4
internaltype DisplayableElement {
  String caption;
  String description;
  String icon;
}

internaltype UiComponent<T> extends DisplayableElement {
  T value;
  []T singleValue;
  T selection;
  []T singleSelection;
  Boolean visible;
  Boolean enabled;
}
```


The PropertyValue Concept (3)



Part III: Conclusion

The Goals



Declarative
Style

A yellow sticky note is pinned to the background with two red pushpins. It features the text 'Declarative Style' in a dark blue, handwritten-style font. A red, rectangular stamp with the word 'APPROVED' in a bold, sans-serif font is placed diagonally over the top right corner of the note.



Visual
Conformity

A yellow sticky note is pinned to the background with two red pushpins. It features the text 'Visual Conformity' in a dark blue, handwritten-style font. A red, rectangular stamp with the word 'APPROVED' in a bold, sans-serif font is placed diagonally over the top right corner of the note.



Extensibility

A yellow sticky note is pinned to the background with two red pushpins. It features the text 'Extensibility' in a dark blue, handwritten-style font. A red, rectangular stamp with the word 'APPROVED' in a bold, sans-serif font is placed diagonally over the top right corner of the note.

The Fine Print



Paradigm
Shift



Data
Centric



Early
Stage

Mission: Impossible

Purely declarative User Interface Modeling

Thank You